UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Um mecanismo para coleta, análise e classificação de tráfego em Redes Definidas por Software utilizando estruturas de *big data*

Dissertação de Mestrado

Alex de Santana Amorim

São Cristóvão – Sergipe

2021

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Alex de Santana Amorim

Um mecanismo para coleta, análise e classificação de tráfego em Redes Definidas por Software utilizando estruturas de *big data*

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof. Dr. Ricardo José Paiva de Britto Salgueiro
Coorientador(a): Prof. Dr. Rubens de Souza Matos Júnior

São Cristóvão – Sergipe

2021

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: ALEX DE SANTANA AMORIM

Em 10 dias do mês de dezembro do ano de dois mil e vinte, com início às 19h00min, realizou-se na Sala virtual https://meet.google.com/oup-vavz-kho. A Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Alex de Santana Amorim**, que desenvolveu o trabalho intitulado: **"Um mecanismo para coleta, análise e classificação de tráfego em Redes Definidas por Software utilizando estruturas de big data.",** sob a orientação do Prof. Dr. **Ricardo José Paiva de Britto Salgueiro.** A Sessão foi presidida pelo Prof. Dr. **Ricardo José Paiva de Britto Salgueiro** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Jean Carlos Teixeira de Araujo** (UFAPE), logo em seguida o Prof Dr **Edward David Moreno Ordonez** (PROCC/UFS) e, em seguida, ao Prof. Dr. **Rubens de Souza Matos Júnior** (PROCC/UFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) aprovado "*(aprovado/reprovado)*". Atendidas as exigências da Instrução Normativa 01/2017/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), Resolução nº 25/2014/CONEPE e da Portaria nº 413 de 27 de maio de 2020 (Banca por videoconferência) que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 10 de dezembro de 2020.

Prof. Dr. Ricardo José Paiva de Britto Salgueiro
(PROCC/UFS)
Presidente

Prof. Dr. Rubens de Souza Matos Junior
(PROCC/UFS)
Examinador Interno

Prof. Dr. Edward David Moreno Ordonez
(PROCC/UFS)
Examinador Interno

Prof. Dr. Jean Carlos Teixeira de Araujo
(UFAPE)
Examinador Externo

ALEX DE SANTANA AMORIM
Candidato

Agradecimentos

À Deus, por me guiar e me manter perseverante até o fim.

Aos meus pais Antônio Cirilo e Maria Inês, pelo exemplo, direcionamento e todo amor.

À minha bela noiva Jamille Andrade. Obrigado por todo amor, compreensão e suporte ao longo de mais esse roteiro que escrevo em minha vida.

Aos meus avós maternos Adeladio (*in memoriam*) e Pureza (*in memoriam*) e aos meus avós paternos Erasmo (*in memoriam*) e Maria, pelo cuidado, dedicação e zelo.

Aos meus irmãos(ãs) Aila, Alécia e Pedro Alef, por sempre estarem presentes.

Às minhas sobrinhas Paula Laianny, Paula Laysa, Lara Vitória e ao meu cunhado Paulo Sérgio, por todo o amor e carinho de sempre.

Ao meu coorientador e colega de trabalho, Prof. Dr. Rubens de Souza Matos Júnior, pelo suporte e direcionamento. Você foi muito importante nesse trajeto.

Em especial, ao meu orientador, Prof. Dr. Ricardo José Paiva de Britto Salgueiro, pela confiança, respeito e profissionalismo dispensado ao longo dessa trajetória.

À Profª Dra. Edilayne Meneses Salgueiro, pelo apoio e discussão crítica do meu trabalho.

Ao amigo Eduardo Fillipe, que acreditou no projeto e não mediu esforços para contribuir na implementação da solução.

Aos amigos José Andeson Morais de Oliveira e Alfredo Menezes Vieira, por todos os diálogos encorajadores e motivadores. Sempre saía renovado após nossas conversas.

Aos professores e técnicos administrativos do programa de Pós-Graduação em Ciência da Computação, pelo aprendizado e convivência valiosa.

Às amizades construídas na Universidade Federal de Sergipe e, em especial, aos amigos do ELAN, pelos momentos de luta e conhecimentos compartilhados.

À minha equipe de trabalho do Instituto Federal de Sergipe, pelo apoio e adequação das minhas atividades diárias.

À equipe da Studio 3T, que acreditou na pesquisa e forneceu uma licença do software Studio 3T for MongoDB.

E a todos que de forma direta ou indireta contribuíram para a realização deste trabalho.

Resumo

Uma variedade de novos aplicativos e serviços surgem a todo momento no universo da Internet, chegam como uma verdadeira avalanche de informações nas infraestruturas de redes e precisam ser compreendidos e tratados em suas particularidades para garantir o bom desempenho e melhor aproveitamento dos recursos da rede. Analisar fluxos e extrair conhecimento para apoiar decisões na administração e intervir em tempo hábil aos possíveis contratempos é um grande desafio para os profissionais de rede em estruturas tradicionais. Diante desse cenário, o conceito de Redes Definidas por Software (SDN - *Software-Defined Networking*) surgiu como uma proposta de abordagem mais dinâmica, gerenciável e adaptável, onde os planos de dados e de controle são desacoplados, permitindo que o controle seja centralizado. Isso torna a tarefa de obter informações do tráfego da rede mais simples e possibilita, inclusive, a utilização desse grande volume de dados (*big data*) na extração de padrões de comportamento do tráfego usando técnicas de aprendizado de máquina. Esses padrões podem ser utilizados para caracterizar e classificar o tráfego e assim aplicar políticas de qualidade de serviço que possam melhorar a utilização dos recursos da rede. Dessa forma, o principal objetivo desse trabalho é desenvolver o DETCCS - *Decision Engine for Traffic Classification and Control in SDN*, um mecanismo de auxílio à tomada de decisões sobre o tráfego em SDN, baseado em: coleta de informações dos fluxos de dados, estruturas de *big data*, identificação de padrões e classes de tráfego e aplicação de políticas para garantir QoS (*Quality of Service*) ou melhorar o aproveitamento da rede. Inicialmente, uma revisão sistemática da literatura foi realizada buscando compreender como o *big data* tem sido explorado para auxiliar a qualidade de serviços em SDN. Esse estudo permitiu assimilar conceitos, identificar ferramentas e elementos que comumente são utilizados na literatura, bem como as lacunas que poderiam ser exploradas para o desenvolvimento desse trabalho. No estudo de caso, os resultados obtidos sobre os dados extraídos de um *dataset* real apontam para uma segregação no comportamento dos fluxos em 3 grupos. Eles demonstram uma distinção quanto a distribuição, volume e duração dos fluxos dentro do cenário analisado. A partir da classificação dos diferentes grupos de fluxo, o DETCCS foi utilizado para aplicação de regras que podiam limitar o tráfego para alguns grupos, em detrimento de outro que hipoteticamente teria maior prioridade. Os experimentos realizados mostram um grande potencial do mecanismo proposto, que pode ser aproveitado por administradores de redes.

Palavras-chave: SDN, QoS, *big data*, aprendizado de máquina, engenharia de tráfego.

Abstract

A variety of new applications and services appear all the time in the Internet universe, they arrive as a veritable avalanche of information in network infrastructures and need to be understood and treated in their particularities to guarantee the good performance and better use of network resources. Analyzing flows and extracting knowledge to support management decisions and intervene in a timely manner to possible setbacks is a major challenge for network professionals in traditional structures. Given this scenario, the concept of Software Defined Networks (SDN) emerged as a proposal for a more dynamic, manageable and adaptable approach, where data plan and control plan are decoupled, allowing centralized control. This makes the task of obtaining information about network traffic simpler and even makes it possible to use this large volume of data (big data) to extract traffic behavior patterns using machine learning techniques. These standards can be used to characterize and classify traffic and thus apply quality of service policies that can improve the use of network resources. Thus, the main objective of this work is to develop the DETCCS - Decision Engine for Traffic Classification and Control in SDN, a mechanism to assist decision making about traffic in SDN, based on the collection of information from data flows, structures big date, identification of patterns and classes of traffic and application of policies to guarantee QoS (Quality of Service) or improve network utilization. Initially, a systematic review of the literature was carried out in order to understand how big data has been explored to assist the quality of services in SDN. This study allowed to assimilate concepts, identify tools and elements that are commonly used in the literature, as well as the gaps that could be explored for the development of this work. In the case study, the results obtained on the data extracted from a real dataset, point to a segregation in the behavior of flows in 3 groups. They demonstrate a distinction as to the distribution, volume and duration of flows within the analyzed scenario. From the classification of the different flow groups, DETCCS was used to apply rules that could limit traffic to some groups, to the detriment of another that would hypothetically have higher priority. The experiments carried out show a great potential of the proposed mechanism, which can be used by network administrators.

Keywords: SDN, QoS, *big data*, machine learning, traffic engineering.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACDPA	*Advanced Control Distributed Processing Architecture*
AN	*Active Networks*
API	*Application Programming Interface*
ATM	*Asynchronous Transfer Mode*
BDMS	*Big Data Management System*
CFTV	Circuito fechado de televisão
CoS	*Class of Service*
CPU	*Central Processing Unit*
CSV	*Extitcomma-Separated-Values*
D-ITG	*Distributed Internet Traffic Generator*
DB	*Data Base*
DCAN	*Devolved Control of ATM Networks*
DETCCS	*Decision Engine for Traffic Classification and Control in SDN*
DICOM	*Digital Imaging and Communications in Medicine*
DiffServ	*Differentiated Services*
DPI	*Deep Packet Inspection*
DSCP	*Differentiated Services Code Point*
ELAN	*Experimental Laboratory on Computer Networks*
HDFS	*Hadoop Distributed File System*
IANA	*Internet Assigned Numbers Authority*
ICMP	*Internet Control Message Protocol*
IETF	*Internet Engineering Task Force*
IntServ	*Integrated Services*
IP	*Internet Protocol*

IPv4	*Internet Protocol version 4*
IPv6	*Internet Protocol version 6*
IRTF	*Internet Research Task Force*
IRTF	*Internet Research Task Force*
ISO	*International Organization for Standardization*
JSON	*JavaScript Object Notation*
KDD	*Knowledge Discovery in Databases*
KPI	*Key Performance Indicators*
MAC	*Media Access Control*
MIB	*Management information base*
ML	*Machine Learning*
MPLS	*MultiProtocol Label Switching*
NETCONF	*Network Configuration*
NFV	*Network FunctionsVirtualization*
NSFNET	*National Science Foundation Network*
OF-CONFIG	*OpenFlow Configuration Protocol*
ONF	*Open Network Foundation*
OPENSIG	*Open Signaling*
OVS	*Open vSwitch*
OVSDB	*Open vSwitch Database Management Protocol*
PC	*Personal Computer*
PHB	*Per Hop Behavior*
QoS	*Quality of Service*
QP	Questão de Pesquisa
RAM	*Random Access Memory*
RESV	*Reservation-request*

RSVP	*Resource reSerVation Protocol*
SDN	*Software Defined Networking*
SLA	*Service Level Agreement*
SNMP	*Simple Network Management Protocol*
SQL	*Standard Query Language*
SVM	*Support Vector Machines*
TC	*Traffic Conditioning*
TCP	*Transmission Control Protocol*
TIC	Tecnologias de Comunicação e Informação
UFS	Universidade Federal de Sergipe
VoIP	*Voice over Internet Protocol*

Sumário

1 Introdução

Com o progresso da Internet, novos tipos de aplicativos e serviços de rede surgiram para os usuários finais, a exemplo da navegação na web, das mensagens de texto, ligações VoIP, e-mail, áudio, videoconferência, jogos on-line, comércio eletrônico e, mais recentemente, dados de redes de sensores (KARAKUS; DURRESI, 2017). Essas novas fontes de dados aparecem como uma verdadeira avalanche de informações, gerando seus próprios fluxos específicos, que precisam ser transmitidos através da Internet. É preciso entender as particularidades de cada fluxo, para que diferentes tratamentos sejam aplicados e assim tornar a entrega bem-sucedida. O que ocorre, por exemplo, nos casos de algumas aplicações que exigem uma certa largura de banda, enquanto outras são mais sensíveis ao atraso.

Analisar fluxos e obter *insight* para apoiar decisões na administração e intervir em tempo hábil aos possíveis contratempos é um grande desafio para os profissionais de rede em estruturas tradicionais. Por vezes, é uma tarefa que se esbarra nas limitações tecnológicas. A análise de dados massivos remete à definição de *big data*, que é relativamente nova e abstrata e, em linhas gerais, representa conjuntos de dados tão grandes e complexos que as ferramentas tradicionais de gerenciamento de dados ou métodos de processamento são inadequados para lidar. A partir desse ponto, torna-se importante ter conhecimento acerca do conceito de *big data*, que é popularmente caracterizado por “5Vs”: Volume (tamanho do conjunto de dados), Variedade (faixa de tipos e fontes de dados), Velocidade (velocidade de geração e alteração de dados de entrada e saída), Valor (utilidade dos dados) e Veracidade (qualidade dos dados) (CUI; YU; YAN, 2016). Deste modo, *big data* representa um avanço na capacidade de armazenamento, manipulação de dados e ferramentas de processamento. Isso implica que análises de dados podem ser realizadas em tempo real ou quase em tempo real (KUNE et al., 2016).

Diante das limitações impostas pelas redes tradicionais, a exemplo de complexidade, escalabilidade, políticas inconsistentes, custo e segurança, surge o paradigma SDN (*Software Defined Networking*). Rede Definida por Software é um paradigma de rede emergente que gera

uma expectativa para melhorar limitações existentes nas infraestruturas de redes tradicionais. Em resumo, cria-se uma ruptura na integração vertical, separando a lógica do plano controle com o plano de dados dos dispositivos de redes (roteadores e *switches*). Com essa separação, os equipamentos tornam-se simples dispositivos de encaminhamento e a lógica de controle é implementada em um controlador, também chamado de sistema operacional de rede. O que permite obter uma visão global centralizada da rede e aumenta a capacidade de programação, otimizando recursos em menos tempo por meio de de dinâmicas e programas SDN automatizados (KREUTZ et al., 2015; KARAKUS; DURRESI, 2017).

Com a visão global da rede, o controlador logicamente centralizado pode obter dados grandes de todas as camadas diferentes (isto é, de camadas físicas para camadas de aplicações) com granularidade arbitrária. Com isso, torna-se viável a utilização de *big data* empregando métodos analíticos para obter percepções de dados, orientar decisões e poder ajudar no projeto e nas operações da SDN (CUI; YU; YAN, 2016).

A classificação de tráfego e a análise estatística de tendências são essenciais para projetar soluções de engenharia de tráfego e gerenciamento de recursos eficiente em uma rede de computadores (BAKHSHI; GHITA, 2015). Conhecer as características do tráfego para auxiliar na administração da rede não é uma necessidade apenas dos cenários atuais, o trabalho de Claffy (1995) já chamava atenção para essa ânsia, que perdura e é alvo de muitas pesquisas científicas até os tempos atuais.

A adoção de técnicas de aprendizado de máquina em grande volume de dados com a finalidade de prever problemas futuros, analisar padrões de dados, identificar e indicar soluções efetivas para problemas em diversas áreas de negócios é uma realidade para cerca de 40% dos líderes de análise de dados globais, segundo artigo publicado por Lobitsky (2017). Essa é uma tendência que deve impulsionar, inclusive, a área de redes de computadores com a adoção da SDN. Trabalhos como os de Desai e S (2015), Jain et al. (2016), BAKHSHI e GHITA (2015) e Kuranage, Piamrat e Hamma (2019) são bons exemplos dessa nova onda.

1.1 Problemática

Tendo em vista as limitações impostas pelas redes tradicionais citadas por KREUTZ et al. (2015) e KARAKUS e DURRESI (2017), bem como o surgimento de diversos serviços e aplicativos que exigem do administrador da rede entender suas particularidades para melhor gerenciar os recursos de redes, o paradigma de Redes Definidas por Software tem se tornado alvo crescente de pesquisas (TRENDS, 2020).

A classificação de tráfego é vista por BAKHSHI e GHITA (2015) e Kuranage, Piamrat e Hamma (2019) como essencial para projetar soluções de engenharia de tráfego e gerenciamento de recursos. Em paralelo a esses avanços das redes de computadores, Lobitsky (2017) aponta para aceitação e adoção de técnicas de aprendizado de máquina em análises de grande volume

de dados em diversas áreas de negócios como indicadores para soluções efetivas.

No contexto de redes de computadores, Amaral et al. (2016) e Fan e Liu (2017) ensinam que a classificação de fluxo baseada em aprendizado de máquina pode superar algumas das limitações de abordagens baseadas em porta e carga útil, que são as mais utilizadas nas redes tradicionais, pois utilizam estatísticas de tráfego independente do protocolo do aplicativo.

Já Desai, S e T (2015), exploram o conceito de *big data* para desenvolver uma arquitetura de processamento distribuído que propõe melhorias nas decisões relacionadas à qualidade de serviços da rede SDN. Diante dessas situações, a solução proposta por esse trabalho explora duas tendências de mercado, *big data* e aprendizado de máquina, em temas como caracterização, classificação de tráfego e qualidade de serviços em Redes de Computares Definidas por Software.

1.2 Justificativa

Avaliar as características de uma rede de computadores é uma atividade complexa e que requer a coleta precisa e oportuna de grandes volumes de dados sobre o comportamento da rede (JAIN et al., 2016). Exige que os administradores possuam informações detalhadas para um eficaz planejamento de capacidade e expansão, e também para definir requisitos de qualidade de serviço como balanceamento de carga, controle de acesso e priorização do tráfego (DESAI; S, 2015).

As Redes Definidas por Software trazem uma nova abordagem para o contexto das redes de computadores com a separação entre o plano de dados e o plano de controle (KHATER; HASHEMI, 2018). Todavia, a Qualidade de Serviço, que já é um tema bastante discutido em redes convencionais, encontra-se em fase inicial em SDN e ainda há muito para evoluir (WALLNER, 2013). Combinar recursos de SDN com tecnologias de *big data* pode trazer benefícios notáveis para diversas áreas, a exemplo das áreas de engenharia de tráfego, design de camada cruzada e segurança (CUI; YU; YAN, 2016; ALWASEL et al., 2017).

A engenharia de tráfego pode ser beneficiada pela SDN e pela análise de *big data* de maneira vantajosa de modo a melhorar o desempenho da rede em larga escala. Por meio de de um controlador centralizado, a SDN facilita a tarefa de obter informações de tráfego e falhas da rede e com o *big data* é possível aplicar técnicas de aprendizado de máquina como algoritmos de predição, de agrupamento, entre outros. E, ainda, converter os resultados em regras ou políticas nas tabelas de fluxos dos comutadores (CUI; YU; YAN, 2016; DESAI; NAGEGOWDA; NINIKRISHNA, 2016).

A investigação na literatura para compreender como esses assuntos estão relacionados e como se beneficiam, foi realizada por meio de uma revisão sistemática buscando entender como o *big data* pode ser explorado para auxiliar o QoS(*Quality of Service*) em Redes Definidas por Software, discutida no Capítulo 3. A revisão sistemática revela indícios que reforçam o que foi

abordado em Cui, Yu e Yan (2016), quando relata que o *big data* pode beneficiar a engenharia de tráfego na SDN. Na revisão, dentre outros tópicos, foi evidenciado que há grande interesse por parte dos pesquisadores em criar soluções capazes de realizar classificação de tráfego e gestão de fluxo em Redes Definidas por Software.

Classificar o tráfego e definir classes para garantir acordos de níveis de serviço é um assunto abordado pelo modelo de Serviços Diferenciados ou DiffServ. Este é um conceito definido pela IETF e discutido em NICHOLS et al. (1998) e Babiar et al. (2006) com intuito de ser soberano ao "melhor esforço", método padrão de entrega de dados em redes IPs. Nesse tipo de serviço, os chamados roteadores de bordas de um domínio DiffServ realizam uma marcação no campo DSCP nos pacotes IPv4 ou IPv6 de determinados fluxos, que são tratados de forma específica pelos roteadores de núcleo da rede, visando a garantia da qualidade de serviço (PONNAPPAN et al., 2002). É possível trabalhar em uma rede SDN através do protocolo OpenFlow, definir classes e tratar essas classes de maneira específicas em toda a rede (WALLNER, 2013).

Com a possibilidade de, por meio de um controlador central, obter-se uma visão global da rede, coletar informações precisas e oportunas de grandes volumes de dados e interagir em tempo de real com os dispositivos de comutação, a SDN se mostra favorável ao gerenciamento dinâmico e eficaz de QoS em uma solução que utilize estruturas de *big data* e aprendizado de máquina para analisar e compreender os padrões de tráfego de rede.

1.3 Objetivos

O objetivo geral desse trabalho é criar um mecanismo, aqui chamado de DETCCS - *Decision Engine for Traffic Classification and Control in SDN*, de auxílio à tomada de decisões sobre o tráfego em SDN, baseado em: coleta de informações dos fluxos de dados, estruturas de *big data*, identificação de padrões e classes de tráfego e aplicação de políticas para garantir QoS ou maximizar o aproveitamento da rede.

Além do objetivo geral, foram estabelecidos os seguintes objetivos específicos, necessários para atingir o objetivo proposto:

- Desenvolver um módulo no controlador de rede para coletar dados estatísticos dos fluxos da SDN;
- Prover um mecanismo para armazenar estatísticas de fluxo da rede em tempo real em estrutura de *big data*;
- Prover um mecanismo para classificar o tráfego da rede em uma estrutura de *big data* utilizando algoritmo de agrupamento;
- Desenvolver um módulo no controlador para aplicar políticas que limitem a taxa de transferência em classes de fluxos nos comutadores de redes;

- Avaliar a arquitetura por meio de estudos de caso em prova de conceito.

1.4 Trabalhos Correlatos

O trabalho de Jose, Nair e Paul (2017) aborda um tema bastante discutido e muito relevante na atualidade: Mineração de dados aplicada em Redes Definidas por Software. No ponto de vista dos autores, estruturas de processamento de fluxo distribuído em tempo real, como Apache Spark, Storm e Samza, podem ajudar a SDN a tomar decisões inteligentes na rede. No entanto, o trabalho está limitado a discussões sobre o assunto e não apresenta soluções práticas que utilizem tais ferramentas.

O trabalho de Semenciuc et al. (2016) coleta informações da rede e armazena no banco Cassandra DB. Por meio de um controlador central e com visão global da rede, são utilizadas implementações em Java usando Spark Framework. Essa combinação forma um mecanismo para estimar a melhor rota de ponta a ponta com base nas informações coletadas durante o processo de análise atual e em fases anteriores. Para isso, utiliza-se dois algoritmos de aprendizado de máquina: Rede Bayesiana e Lógica Fuzzy. O trabalho é de grande contribuição para engenharia de tráfego, no entanto está focado em encontrar o melhor caminho de ponta a ponta.

Os trabalhos Desai, S e T (2015), Desai, Nagegowda e Ninikrishna (2016), Desai e S (2015) são uma sequência de pesquisas dos mesmos autores, onde relatam discussões ou resultados dos experimentos realizados em uma nova arquitetura proposta, combinando artifícios de QoS, *big data* e SDN.

Em Desai, S e T (2015), uma nova arquitetura é proposta. Denominada de ACDPA (*Advanced Control Distributed Processing Architecture*), foi implementada utilizando a estrutura Hadoop para processamento de grandes volumes de dados e o OpenDaylight como controlador SDN.

O ACDPA funciona da seguinte forma: utiliza a ferramenta Mininet para simular a rede SDN; captura pacotes de switches designados utilizando o *sniffer* de rede wireshark; realiza um pré-processamento para manter as informações necessárias para classificar os pacotes; o Hadoop é usado para processar os pacotes capturados pelo *sniffer*; os pacotes são classificados em 4 parâmetros (Tamanho, Endereço de origem, Endereço de destino, Tipo de pacote); o OpenDaylight é usado para controlar o plano de dados da rede através do *feedback* fornecido pela estrutura Hadoop.

É importante destacar que a única ação do controlador na rede é a priorização dos fluxos com base em um dos parâmetros de classificação, a exemplo do endereço de origem. O trabalho não demonstra flexibilização em relação a esses parâmetros. Além disso, a captura dos pacotes é realizada por uma ferramenta *sniffer* e não é explicado se o *log* resultante é fornecido de forma automática ou manual ao Hadoop. Em relação à classificação do fluxo, não são utilizadas técnicas

de aprendizado de máquina ou serviço diferenciado, embora seja um desejo de trabalho futuro.

O trabalho Desai, Nagegowda e Ninikrishna (2016) é uma continuação de (DESAI; S; T, 2015), onde é discutido melhorias na segurança e também na arquitetura. Em relação à segurança, foram reproduzidos os testes realizados em Mantur, Desai e Nagegowda (2015) e o experimento mostrou que é possível identificar ataques através do aumento de pacotes ICMP para um destino, tal como, ataques de negação de serviço distribuídos ou vindos de uma única origem. Sobre a arquitetura, as mudanças ficam limitadas a discussões de possíveis implementações de algoritmos de agrupamento no Hadoop e de como aplicar qualidade do serviço em cada categorias de fluxo, como por exemplo, reservar largura de banda para cada categoria.

Para finalizar a sequência, o trabalho Desai e S (2015) inclui o conceito de nuvem em sua arquitetura, reforça a segurança com o Kerberos e adiciona um componente muito interessante, o balanceador de cargas inteligentes. O balanceador funciona como distribuidor de pacotes nos comutadores, além de receber do controlador informações de controle para enviar informações dos pacotes ao Hadoop. Esse trabalho fica limitado às discussões e não menciona testes para validar sua funcionalidade.

O estudo de Jain et al. (2016) mostra como a análise de grandes volumes de dados pode ser utilizada para melhorar o gerenciamento de QoS em uma SDN. Para tal, realizam-se análises multidimensionais dos principais indicadores de desempenho (KPIs - *Key Performance Indicators*). A pesquisa utiliza ambiente simulado e ferramentas como D-ITG e IPerf3 para geração de tráfego, Mininet para emular a SDN e o controlador de código aberto POX. Além disso, foi desenvolvido um módulo de análise em java para converter os valores de MIB (*Management information base*) em arquivos CSV (textitcomma-separated-values). O arquivo é integrado com a ferramenta "R"para manipulação, mineração, cálculo e exibição de dados. Durante as simulações, 24 KPIs foram coletadas, dentre elas, estão: bytes - enviados e recebidos; pacotes - enviados, recebidos; perda de pacotes - entrada, saída; CPU - controlador, máquina virtual; RAM - usada, em buffer, em cache; pacote - taxa, tamanho, perda; fluxos - atraso, instabilidade, taxa de transferência.

Os dados coletados foram analisados pelo algoritmo Spearman para correlacionar dados de séries temporais para cada KPI. Além desse algoritmo, foi utilizado o M5Rule. Ele é um algoritmo de aprendizado de máquina de regressão linear utilizado para combinar árvores de decisão e regressão linear para previsão. Esse mecanismo foi utilizado para qualificar KPI e identificar qual teve maior impacto geral na QoS.

O trabalho resultou em constatações que foram classificadas em três grupos, são eles: correlações esperadas, descobertas e inesperadas. As constatações esperadas mostram resultados óbvios, como, por exemplo, o *jitter* está correlacionado com o atraso em um coeficiente de Spearman de +0,95. As constatações descobertas fornecem informações de desempenho, por exemplo, A CPU está correlacionada com o número de pacotes transmitidos nas filas. Já com as constatações inesperadas, descobriu-se que altas taxas de transmissão na fila estavam

correlacionadas com o atraso.

O trabalho é bastante interessante, no entanto, não trás nenhuma dinamicidade com a SDN. A solução proposta resulta em uma realização da captura dos dados e uso de ferramentas que precisam de interação humana e, geralmente, não são de fácil uso; além disso, não realiza intervenção na rede após obter os resultados de análise.

Por fim, o trabalho de Alwasel et al. (2017) sugere o desenvolvimento de uma arquitetura de programação holística para tornar os BDMSs(*Big Data Management System*) nativos em SDN. A arquitetura proposta está concentrada em três pontos principais: gerenciamento de configuração, agendamento e gerenciamento de tráfego. De forma objetiva, a intenção é otimizar conjuntamente o desempenho de aplicativos de *big data* e o desempenho da SDN em tempo de execução. O trabalho não fez nenhum tipo de experimento e está limitado à proposta e a discussão sobre o assunto.

Os trabalhos a seguir não entraram na revisão sistemática devido a *string* de busca não contemplar o termo "*machine learning* ou eles não possuírem o termo "QoS"ou "*Quality of Service*". Inicialmente, aprendizado de máquina não fazia parte do planejamento desse estudo, embora tenha se mostrado um termo bastante comum nos resultados da revisão sistemática. Desta forma, eles contribuíram com muita relevância para o desenvolvimento dessa dissertação.

O trabalho de BAKHSHI e GHITA (2015) tem o objetivo de encontrar o perfil de tráfego do usuário com base nas tendências de uso de aplicativos. Para isso, usa dados coletados através do NetFlow em uma rede de computadores com cerca de 250 usuários. Nesse trabalho, o algoritmo de agrupamento k-means foi utilizado para encontrar esses perfis. Embora o trabalho enfatize as possibilidades de trabalhar os resultados obtidos com SDN, isso não foi demonstrado de forma prática, apenas discussões foram feitas sobre o assunto.

O artigo de Kuranage, Piamrat e Hamma (2019) foi publicado após a conclusão da revisão sistemática e também utilizou o aprendizado de máquina para classificar tráfegos em redes SDN, inclusive, usou o algoritmo k-means para entender o comportamento do tráfego e posteriormente rotular os grupos em classes. Além disso, esse trabalho utilizou o resultado do k-means para criar modelos de algoritmos de aprendizado de máquina supervisionado e compará-los. De acordo com o estudo, o algoritmo supervisionado *Support Vector Machines*,SVM (linear), teve uma melhor precisão na classificação supervisionada. No entanto, o artigo não cita o desenvolvimento de nenhum mecanismo para armazenar os dados capturados da rede, ele utiliza uma base pronta para as análises realizadas no trabalho.

Por fim, o trabalho de Le et al. (2018) propõe uma nova arquitetura que pretende integrar vários algoritmos de aprendizado de máquina, *big data*, SDN, NFV(*Network Functions Virtualization*), redes auto-organizadas e divisão da rede em tecnologia 5G. Em seu experimento, utiliza o algoritmo de aprendizado de máquina não supervisionado k-means para agrupar fluxos que tenham comportamento de tráfego semelhantes. Os grupos resultantes são rotulados em

classes que são associadas às fatias criadas pelo administrador. Essas fatias limitam o tráfego da rede usando entradas de fluxos e medidores através do OpenFlow.

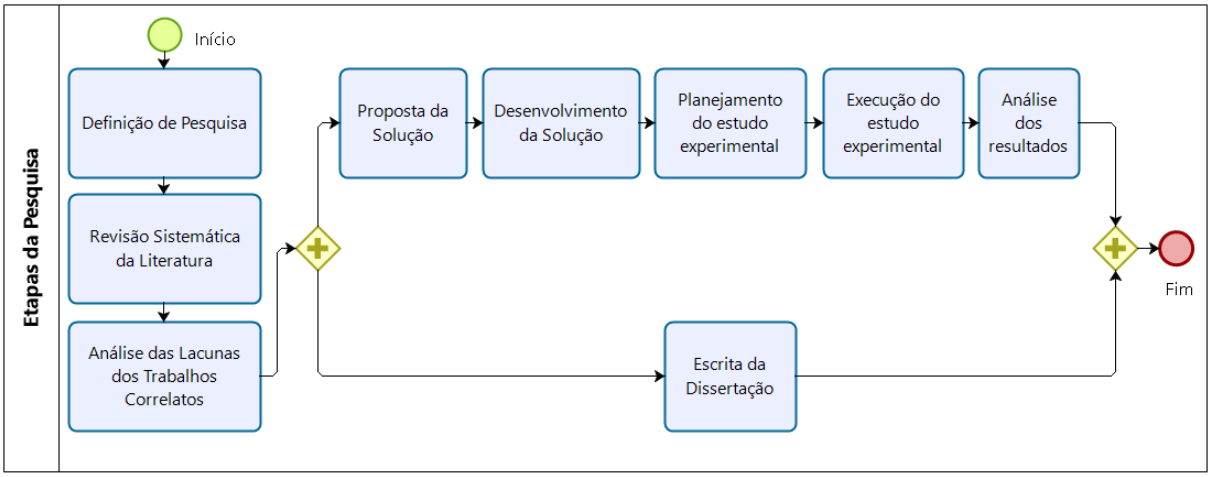
1.5 Metodologia

A palavra metodologia vem do grego "meta"= ao largo; “odos” = caminho; “logos” = discurso, estudo. É compreendida como uma disciplina que consiste em estudar, compreender e avaliar os vários métodos disponíveis para a realização de uma pesquisa. Em outras palavras, é a aplicação de procedimentos e técnicas que devem ser observados para construção do conhecimento, com o propósito de comprovar sua validade e utilidade nos diversos âmbitos da sociedade (PRODANOV; FREITAS, 2013).

1.5.1 Etapas da Pesquisa

O planejamento dessa pesquisa seguiu algumas etapas importantes, ilustradas na figura 1, para que fosse bem conduzida. Foram 8 atividades desenvolvidas sequencialmente e 1 atividade desenvolvida paralelamente a partir da terceira. As etapas são descritas a seguir:

Figura 1 – Etapas da Pesquisa



Fonte: Autor (2019)

1. **Definição da pesquisa:** a leitura do artigo Cui, Yu e Yan (2016) despertou o interesse pelas possibilidades que duas grande áreas, *big data* e SDN, podem proporcionar trabalhando em conjunto. Inicialmente, a ideia era pesquisar sobre Qualidade de Serviço (QoS) em SDN.

2. **Revisão Sistemática da Literatura:** essa etapa foi fundamental para identificar os trabalhos existentes na literatura sobre o tema. Para isso, foram realizadas pesquisas em bases

de artigos científicos, teses, dissertações e demais. Essa etapa contribuiu bastante para assimilar os conceitos teóricos.

3. **Análise das lacunas dos trabalhos correlatos:** nessa fase, buscou-se identificar os pontos positivos e negativos nos trabalhos resultantes da revisão sistemática da literatura. Essa etapa foi fundamental para guiar os caminhos a serem seguidos.

4. **Proposta da solução:** o objetivo dessa atividade foi propor uma nova solução que permita coletar informações dos fluxos dos dispositivos de comutação através do controlador de rede e faça uso de estruturas de *big data* para armazenar, analisar e compreender os padrões de tráfego de rede, identificar classes de tráfego e aplicar políticas que contribuam com a garantia na qualidade do serviço e maximize o aproveitamento dos recursos de redes multiserviços.

5. **Desenvolvimento da solução:** essa etapa consistiu na busca de soluções que se integrem e formem, juntamente com o desenvolvimento de módulos no controlador de rede SDN, um mecanismo que permita a coleta de estatísticas dos fluxos, análise e processamento de grandes volumes de dados em estruturas de *big data*, resultando na solução DETCCS, que será explanada ao longo desse trabalho.

6. **Planejamento dos estudos de caso:** um planejamento inicial para realizar os estudos de caso foi realizado e é explanado na Seção 5.1.

7. **Execução dos estudos de caso:** essa etapa é, na prática, a execução do planejamento dos estudos de caso. Ela só foi possível após a conclusão do desenvolvimento da solução proposta (DETCCS) e consequentemente do planejamento do estudo de caso.

8. **Interpretação dos resultados:** o objetivo é apresentar uma análise descritiva dos resultados obtidos com a execução do experimento.

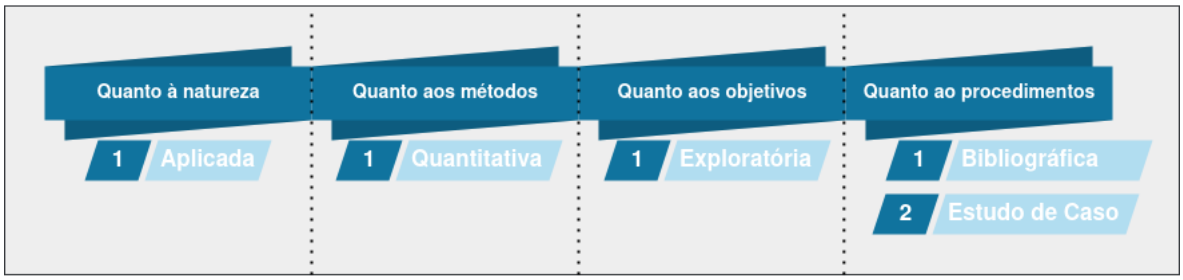
9. **Escrita da Dissertação:** essa etapa consiste na elaboração da escrita da dissertação. Aqui, pretende-se detalhar toda e qualquer contribuição que o trabalho venha a oferecer, de forma que possa ser reproduzido por demais pesquisadores ou afins.

1.5.2 Classificação da Pesquisa

A metodologia pode ser diferenciada em quatro tipos: no que diz respeito à natureza, aos métodos, aos objetivos e aos procedimentos (NASCIMENTO; SOUSA, 2017). A figura 2 ilustra a forma como este trabalho está classificado mediante a esses quatro pontos.

Este trabalho foi desenvolvido por meio de um ponto de vista metodológico de natureza aplicada, uma vez que a pesquisa aplicada é dedicada à geração de conhecimento para solução de problemas específicos e é dirigida à busca da verdade para determinada aplicação prática em situação particular (NASCIMENTO; SOUSA, 2017).

Figura 2 – Metodologia de Pesquisa.



Fonte: Autor (2019)

No que diz respeito aos objetivos, pode-se dizer que essa pesquisa está classificada como pesquisa de natureza exploratória. Gil (2002) explica que esse tipo de pesquisa procura proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito ou a constituir hipóteses, para aprimorar ideias ou descobrir intuições.

Do ponto de vista dos métodos (ou abordagens metodológicas), o trabalho pode ser classificado como pesquisa quantitativa. Para (NASCIMENTO; SOUSA, 2017), pesquisas quantitativas empregam medidas padronizadas e sistemáticas, reunindo respostas pré-determinadas, facilitando a comparação e a análise de medidas estatísticas de dados.

No tocante aos procedimentos técnicos, é possível que a pesquisa esteja em diferentes classificações. Para esse trabalho, foi proposto uma pesquisa de cunho bibliográfico e experimental. É Bibliográfica por fazer uso de fontes secundárias, a exemplo de dissertações, monografias, teses, livros, jornais ou quaisquer outras publicações concedidas por ferramentas acessíveis ao público (MARCONI; LAKATOS, 2003); e é classificada como estudo de caso por buscar validar a solução proposta em forma de provas de conceito, objetivando uma investigação profunda e exaustiva ao analisar fenômenos de um determinado ambiente, por meio de múltiplas fontes de evidências (YIN, 2015).

1.6 Organização da Dissertação

Este trabalho está organizado em 6 (seis) Capítulos. Neste primeiro, foi apresentado o problema de pesquisa, a justificativa para sua realização, os objetivos e sua organização. Os demais Capítulos estão divididos da seguinte forma:

O Capítulo 2 apresenta a fundamentação teórica, que respalda os conceitos fundamentais para realização deste trabalho, tais como: *big data*, redes definidas por software e qualidade de serviço.

O Capítulo 3 apresenta a revisão sistemática da literatura. Esse Capítulo buscou evidências em estudos primários para responder algumas questões de pesquisas e entender como o *big data* pode ser explorado para auxiliar a QoS em Redes Definidas por Software.

No Capítulo 4 é descrita a solução proposta (DETCCS). Nesse Capítulo buscou-se detalhar todos os componentes utilizados no mecanismo bem como suas funcionalidades.

O Capítulo 5 apresenta o planejamento, os objetivos e a execução dos estudos de caso, bem como uma discussão a respeito do experimento e dos resultados obtidos.

Por fim, o Capítulo 6 apresentada as conclusões, as dificuldades, os trabalhos futuros e as contribuições relacionadas à dissertação.

2

Fundamentação Teórica

Este Capítulo aborda os principais conceitos acerca do tema proposto, procurando embasamento na literatura para fortalecer o trabalho. Desta feita, são apresentados os conceitos teóricos relacionados a Redes Definidas por Software, Qualidade de Serviço, *Big Data*, Coleta, Caracterização e Classificação de Tráfego.

2.1 Redes Definidas por Software

Rede definida por software (SDN) é um paradigma de rede emergente que gera uma expectativa para melhorar limitações existentes nas infraestruturas de redes tradicionais. Em suma, cria-se uma ruptura na integração vertical, separando a lógica do plano controle com o plano de dados dos dispositivos de redes (roteadores e *switches*). Com essa separação, os equipamentos tornam-se simples dispositivos de encaminhamento e a lógica de controle é implementada em um controlador, também chamado de sistema operacional de rede (KREUTZ et al., 2015).

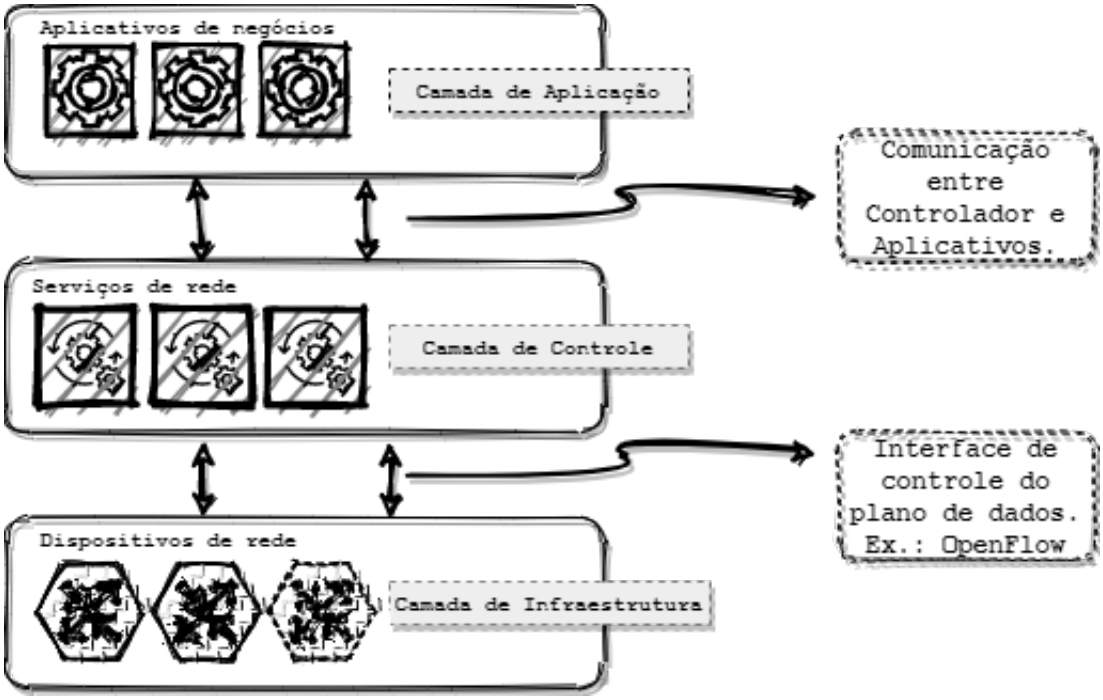
A proposta da SDN simplifica drasticamente a aplicação de políticas e configurações, impulsionando a evolução e a inovação da rede, já que a inteligência de rede é logicamente centralizada em controladores externos, baseados em software, que podem ser implementados com o uso da tecnologia de servidores comerciais (PCs), um recurso abundante, escalável e barato. É uma verdadeira quebra de paradigma, visto que SDN propõe o fim das “caixas pretas” da Internet, ou seja, é o fim das implementações integradas verticalmente baseadas em software fechado sobre hardware proprietário. Desta feita, a definição do comportamento da rede em software pode ser determinada não apenas pelos fabricantes do equipamento, mas também por fornecedores ou pelos próprios usuários, a exemplo dos operadores de rede (NUNES et al., 2014; ROTHENBERG et al., 2010; KREUTZ et al., 2015).

Esse novo conceito em redes de computadores tem ganhado atenção no âmbito acadêmico

e industrial. A *Open Network Foundation*, ONF, é uma organização industrial criada por um grupo de operadores de redes, provedores de serviços e fornecedores com intuito de promover a SDN e padronizar o protocolo *OpenFlow*. No que tange ao âmbito acadêmico, a *OpenFlow Network Research Center* foi criada com foco na pesquisa. Além dessas instituições, a *Internet Engineering Task Force* (IETF) e *Internet Research Task Force* (IRTF), entre outras organizações produtoras de normas, desprendem esforços na padronização da SDN (NUNES et al., 2014).

A figura 3 descreve de forma resumida a visão lógica da arquitetura SDN. A inteligência da rede é logicamente centralizada por meio de controladores baseados em software na camada de controle. Na camada de infraestrutura estão os elementos de encaminhamento (comutadores). As aplicações de negócios, por sua vez, estão na camada de aplicação. A integração entre os elementos de encaminhamento e o controlador é chamada de *southbound*, já a comunicação entre a camada de aplicação e o controlador é conhecida como *northbound*.

Figura 3 – Arquitetura de uma Rede Definida por Software.



Fonte: Adaptado (ONF, 2019)

Embora a atenção recebida pela indústria tenha crescido de forma considerável nos últimos tempos, a ideia de redes programáveis e a lógica de controle desacoplada existe há muitos anos. NUNES et al. (2014) e XIA et al. (2015) apontam as redes ativas (*Active Networks* - AN), em meados da década de 90, como sendo um dos primeiros esforços para promover a capacidade de programação da rede. A partir daí, grupos de trabalhos foram surgindo, a exemplo do *Open Signaling* (OPENSIG), cujo objetivo era tornar as redes ATM mais abertas, extensíveis e programáveis; DCAN (*Devolved Control of ATM Networks*), cuja premissa era que as funções de controle e gerenciamento de muitos dispositivos (comutadores ATM) fossem dissociadas dos

próprios dispositivos e delegadas a entidades externas dedicadas a esse propósito; por sua vez, o Projeto 4D, em 2004, defendeu um design limpo que enfatizasse a separação entre a lógica de decisão de roteamento e os protocolos que governavam a interação entre os elementos da rede. Em 2006 surge a proposta do IETF, o NETCONF, um protocolo de gerenciamento para modificar a configuração de dispositivos de rede, permitindo aos dispositivos dispor de uma API (*Application Programming Interface*) da qual os dados de configuração extensíveis pudessem ser enviados e recuperados. Entre 2006 e 2007 surge o projeto Ethane, considerado o antecessor do OpenFlow, cujo foco consistia em usar um controlador centralizado para gerenciar políticas e segurança em uma rede.

As vantagens da SDN advêm do desacoplamento peculiar entre o plano de controle e o plano de dados, isso permite uma maior gestão na rede por meio da programação. O controle adotado pela SDN vai desde o encaminhamento de pacotes em nível de comutação, até ajustes de links em nível de enlace, quebrando a barreira de camadas. Além disso, permite obter o status de forma instantânea da rede e de forma centralizada controlar em tempo real, levando benefícios na otimização de configuração e consequentemente melhoria no desempenho. O potencial benefício é ainda mais evidente pelo fato da SDN oferecer uma plataforma conveniente para experimentações de novas técnicas e encorajar novos projetos através da programação e da capacidade de definir redes virtuais isoladas através do plano de controle (XIA et al., 2015).

A tabela 1 demonstra de forma resumida essas vantagens em comparação às redes convencionais, conforme explica (XIA et al., 2015).

Tabela 1 – Comparação Entre Redes SDN e Redes Convencionais

	SDN	**Redes convencionais**
Característica	Dados desacoplados, plano de controle e programabilidade.	Um novo protocolo por problema, controle de rede complexo.
Configuração	Configuração automatizada com validação centralizada.	Configuração manual propensa a erros.
Performasse	Controle global dinâmico com informações cruzadas.	Informações limitadas e configuração relativamente estática.
Inovação	Fácil implementação de software para novas ideias, ambiente de teste suficiente com isolamento e rápida execução usando atualização de software.	Implementação de hardware difícil para novas ideias, ambiente de teste limitado, longo processo de padronização.

Fonte: (XIA et al., 2015)

Embora tenha promessas de configuração aprimorada, melhorias no desempenho e incentivo à inovação, a SDN ainda está engatinhando e ainda há muito o que ser feito, sobretudo na padronização e na adoção da tecnologia. Preocupações comuns incluem a interoperabilidade de SDN com dispositivos de rede legados, desempenho e privacidade de controle centralizado e ainda falta de especialistas para suporte técnico. As implementações existentes de SDN são frequentemente limitadas a pequenos testes para protótipos de pesquisa. Os protótipos para fins de pesquisa permanecem prematuros para oferecer confiança para a implantação no mundo real

(XIA et al., 2015).

Considerando os pontos mencionados até aqui, pode-se conceituar a SDN como um paradigma de redes de computadores emergente caracterizada pela existência de um sistema de controle (software) dissociado e que pode gerenciar o mecanismo de encaminhamento dos elementos de comutação da rede por uma interface de programação bem definida. Essa dissociação simplifica e permite a flexibilização e evolução da rede por meio do desenvolvimento de novos serviços por meio de uma interface aberta e bem definida entre os dispositivos e o controlador da rede.

Nas subseções a seguir, os elementos característicos das duas entidades principais da SDN, entidade de controle e entidade de encaminhamento, são apresentados com maior nível de detalhes.

2.1.1 Elementos de Borda

Os elementos de borda, também conhecidos como dispositivos de usuários, são elementos não programáveis (computadores, impressoras, etc) que estão conectados aos dispositivos de rede na camada de infraestrutura de uma SDN.

2.1.2 Dispositivos de Comutação

Os dispositivos de comutação ou elementos de rede programáveis são encaminhadores de mensagens e têm o objetivo de direcionar os pacotes ao destino. A operação de encaminhamento dos elementos de comutação em redes funciona da seguinte forma: o pacote é recebido pela interface, em seguida é inspecionado e logo após a tabela de encaminhamento é consultada para saber qual é o destino do pacote. Se a consulta obtiver sucesso, o pacote é enviado à porta de destino do comutador, caso contrário, o pacote é eliminado ou é realizado um procedimento padrão. É de suma importância destacar que as regras de encaminhamento são definidas no controlador, na camada de controle e enviadas aos dispositivos de comutação das redes por meio de uma interface bem definida, as quais são armazenadas em memória local. Uma dessas interfaces é o OpenFlow que é feito sob medida para a operação dos principais roteadores e *switches* (XIA et al., 2015; MACEDO et al., 2015).

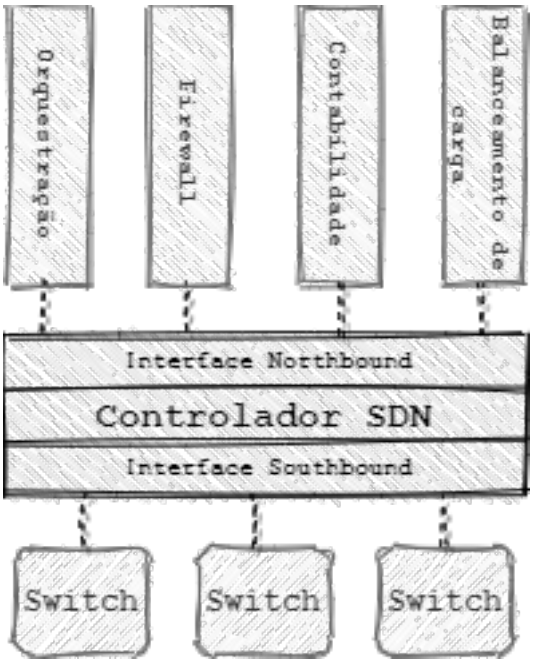
2.1.3 Controlador

O Controlador é o componente mais importante na arquitetura SDN (XIA et al., 2015). Ele é a entidade que monitora e modifica o comportamento dos elementos de redes e que constrói e apresenta um mapa lógico de toda rede. O plano de controle de uma SDN pode ser gerenciado por um ou mais controladores e funciona da seguinte forma: o controlador atua como um sistema operacional de rede e disponibiliza interfaces de programação de alto nível, isso permite que desenvolvedores consigam abstrair detalhes do comportamento de cada componente da rede

e possibilite a criação de programas de controle. O controlador traduz esses programas em ações que podem ser enviadas para cada elemento da rede, normalmente utilizando um conjunto de APIs para interagir de forma segura e garantindo que os comandos de baixo nível sejam executados na ordem correta. A comunicação entre o controlador e os elementos de rede seguem padrões de interfaces que são limitadas ao hardware. Na atualidade, o protocolo OpenFlow tem se destacado nesse sentido (MACEDO et al., 2015).

A figura 4 e a explicação de como acontece a dinâmica da SDN deixam claro que o controlador faz uso de duas interfaces de comunicação. A primeira é responsável pela comunicação com os aplicativos de controle, está ligada ao lado norte do controlador e utiliza linguagem de alto nível, essa interface é chamada de *Northbound*. A segunda, chamada de *Southbound*, lida com transações na camada de infraestrutura, isto é, coletando o status da rede e atualizando as regras de encaminhamento de pacotes para comutar dispositivos na camada de infraestrutura, estando ligada ao lado sul do controlador (XIA et al., 2015).

Figura 4 – Interfaces Southbound e Northbound da SDN



Fonte: Adaptado (XIA et al., 2015)

Esse modelo de rede assume que o controle é centrado. No entanto, uma estrutura SDN não está limitada a um único controlador fisicamente centralizado. Isso representaria um ponto único de falha para toda a rede, e, portanto, uma fragilidade muito grande. Geralmente, os dispositivos que possuem suporte a protocolos compatíveis com a SDN possibilitam vários controladores conectados simultaneamente e, em caso de falhas, controladores de backup assumem o controle da rede (NUNES et al., 2014).

2.1.4 Aplicações de Rede

As aplicações de rede estão acima da camada de controle. Através da camada de controle, os aplicativos SDN podem acessar adequadamente uma visão de rede global com status instantâneo através da interface *northbound* do controlador. De posse dessas informações, os aplicativos SDN podem implementar programaticamente estratégias para manipular as redes físicas subjacentes usando uma linguagem de alto nível, ou seja, podem determinar o comportamento dos dispositivos de rede ao enviar regras de configurações para o plano de dados através do controlador. Dentre várias possibilidades de aplicativos estão: aplicativos para roteamento, limitação de taxa de transferência, balanceamento de carga, manutenção da rede, segurança e etc (XIA et al., 2015).

2.1.5 Interface Northbound

A interface *Northbound* funciona como uma ponte, ligando o controlador SDN às aplicações de rede. A expressão "norte", traduzida para o português, faz uma referência à direção controlador/aplicação cuja comunicação entre os serviços e aplicação devem ser executadas, conforme mostra a figura 4. Uma de suas principais características é ser definida inteiramente em software. Dito isso, é natural que aplicações de gerenciamento externos ou serviços de rede pretendam extrair informações da rede subjacente ou controlar aspectos relacionados ao comportamento e/ou políticas de rede e a API torna isso possível. A *Northbound* utiliza linguagem de programação de alto nível, o que facilita a interação e a criação de novas soluções, já que a programação de baixo nível é abstraída.

Um ponto bastante discutido sobre a API *Northbound* é a padronização. Atualmente não existe um padrão definido para essa interface, no entanto, vários projetos estão seguindo nessa direção (NUNES et al., 2014; MACEDO et al., 2015).

2.1.6 Interface Southbound

Ao contrário da interface *northbound*, a *southbound* é padronizada. É ela que cuida da interação entre o controlador, os elementos de rede e as interfaces programáveis nos elementos de borda. A expressão "sul", traduzida para o português, faz uma referência à direção da comunicação do controlador com os elementos de rede que, nesse caso, está em um nível mais inferior da arquitetura. Um contraste com a *northbound* que faz a ponte entre controlador e aplicações de rede em um nível mais elevado. A figura 4 pode ser observada para uma visualização mais clara sobre a localização da interface *southbound* na arquitetura SDN.

O protocolo OpenFlow é um dos exemplos mais conhecidos de uma API *southbound* no contexto de SDN. Com ele é possível controlar remotamente as tabelas de fluxo em elementos de rede programáveis, ou seja, a comunicação entre os elementos físicos da rede e o controlador torna-se possível por meio desse protocolo (MACEDO et al., 2015).

2.1.7 Virtualizador de Rede

Com a concepção da SDN, se tornou possível identificar padrões por meio de pacotes trafegados pelos comutadores e abriu-se a possibilidade de se associar diversos comportamentos em uma mesma rede. Isso viabilizou, por exemplo, manter um comportamento tradicional para fluxos denominados "de operação" e dar tratamento diferente para fluxos "de pesquisa" em uma rede acadêmica. Essa divisão "virtual" no tratamento de diferentes padrões de fluxo é a função do virtualizador de rede. É dele a responsabilidade de gerenciar as redes virtuais e prover divisões de recursos físicos entre essas redes. Desta forma, cada rede virtual formada pelo divisor pertence a uma visão diferente e possui o próprio sistema operacional de rede. Sendo assim, é possível coexistir diversos controladores funcionando sobre uma mesma rede física, entretanto separados virtualmente (GUEDES et al., 2012).

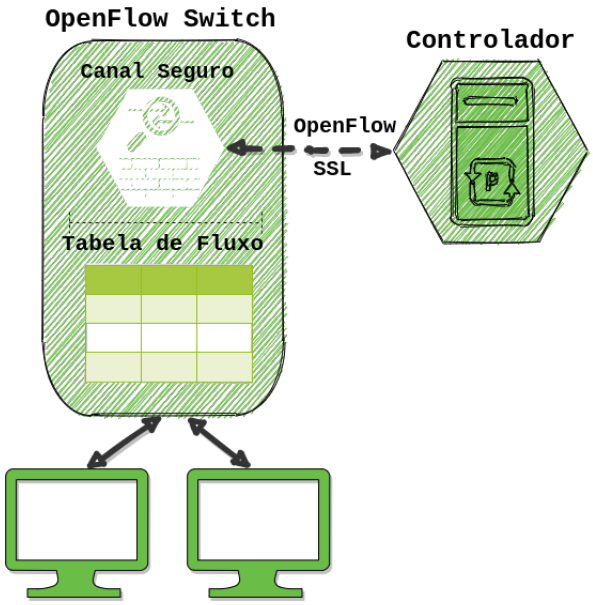
2.1.8 Protocolo OpenFlow

Existem vários padrões de protocolo sobre o uso de SDN em aplicativos reais. Um dos padrões de protocolo mais populares é o OpenFlow (NUNES et al., 2014; HU; HAO; BAO, 2014). O OpenFlow foi proposto na Universidade de Stanford e é considerado o protocolo padrão da SDN. Em seus *testbeds*, muitos projetos de código aberto foram propostos para governar controladores e chaves SDN universais.

A especificação do OpenFlow descreve-o como um protocolo aberto para permitir que aplicativos de software programem a tabela de fluxo de diferentes *switches* (FOUNDATION, 2012). Impulsionado pelo princípio da SDN de desacoplar os planos de controle e encaminhamento de dados, o OpenFlow padroniza a troca de informações entre os dois planos por meio de uma interface de programação bem definida e com a capacidade de controlar, à distancia, a tarefa de encaminhamento de pacotes (MCKEOWN et al., 2008; NUNES et al., 2014). É um protocolo orientado por fluxo e possui abstração de *switches* e portas para controlar o fluxo. Com relação aos *switches*, o OpenVSwitch (OVS) (OPENVSWITCH, 2019) é um dos mais populares, seu kernel é escrito em Linux 3.3 e seu *firmware* está disponível em Pica8 e Indigo (HU; HAO; BAO, 2014).

Uma arquitetura OpenFlow é composta por três componentes principais, como demostrado e ilustrado na Figura 5: um *switch* compatível com OpenFlow, um canal seguro e um controlador. Os *switches* usam tabelas de fluxo para encaminhar pacotes. O controlador é um programa de software responsável por manipular a tabela de fluxo do *switch* usando o protocolo OpenFlow. O canal seguro é a interface que conecta o controlador a todos os *switches*. Por meio deste canal, o controlador gerencia, recebe e envia pacotes (LARA; KOLASANI; RAMAMURTHY, 2014).

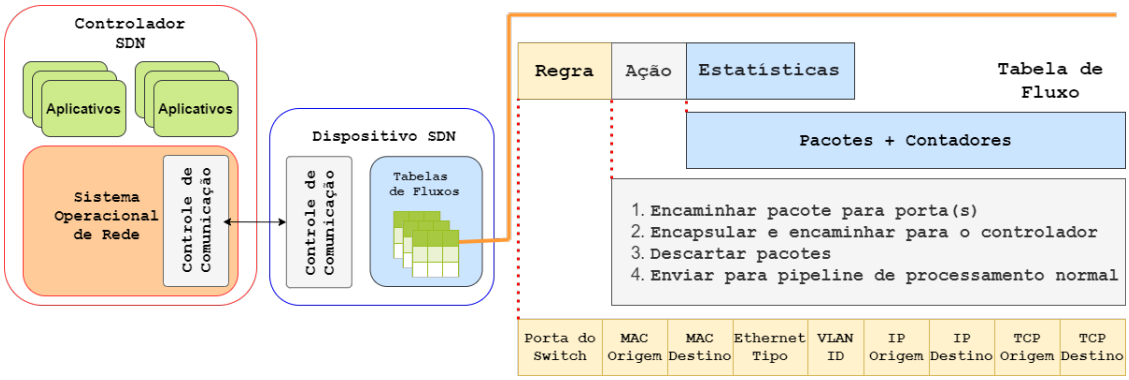
Figura 5 – Principais Componentes do OpenFlow



Fonte: Adaptado (MCKEOWN et al., 2008)

Na chegada de um pacote a um *switch* OpenFlow, os campos de cabeçalho de pacote são extraídos e comparados com a parte dos campos correspondentes das entradas da tabela de fluxo. Se uma entrada correspondente for encontrada, o comutador aplicará um conjunto apropriado de instruções ou ações associadas à entrada de fluxo correspondente. Se o procedimento de consulta da tabela de fluxo não resultar em uma correspondência, a ação executada pelo comutador dependerá das instruções definidas pelo erro de tabela definido para entrada de fluxos. Toda tabela deve conter uma entrada para lidar com erros. Essa entrada especifica um conjunto de ações a serem executadas quando nenhuma correspondência for encontrada para um pacote de entrada, como descartar o pacote, continuar o processo de correspondência na próxima tabela de fluxo ou encaminhar o pacote para o controlador através do canal OpenFlow, como demonstrado e ilustrado na figura 6 (NUNES et al., 2014).

Figura 6 – Dispositivo SDN Habilitados para OpenFlow.



Fonte: Adaptado (KREUTZ et al., 2015)

Atualmente, as especificações do protocolo OpenFlow são mantidas pela *Open Network Foundation* - (ONF), uma entidade aberta formada por pesquisadores de universidades, administradores de rede, agências do governo e interessados em geral. Em seu site, a ONF destaca, dentre muitos parceiros, empresas como Google, Samsung, Dell e Intel, consideradas gigantes da tecnologia nos tempos atuais (ONF, 2019).

O OpenFlow suporta três tipos de mensagens: controlador para comutador, assíncrono e simétrico, cada um com vários subtipos. As mensagens do controlador para comutador são iniciadas pelo controlador e podem ou não exigir uma resposta do interruptor. As mensagens assíncronas são iniciadas pelo comutador e usadas para atualizar o controlador de eventos de rede e alterações no estado do comutador. Por fim, as mensagens simétricas são enviadas sem solicitação, em qualquer direção (FOUNDATION, 2012).

As mensagens assíncronas são utilizadas nesse trabalho, portanto, é de suma importância a compreensão sobre seus subtipos, quais sejam:

- ***Packet-in*** - transfere o controle de um pacote para o controlador;
- ***Flow-Removed*** - informe ao controlador que o fluxo foi removido;
- ***Port Status*** - informe o controlador sobre uma alteração em uma porta do *switch*;
- ***Error*** - o *switch* pode notificar o controlador sobre problemas usando mensagens de erro.

Outro componente importante no OpenFlow é a tabela de fluxo. Ela é parte fundamental de um *switch openflow*. Essas tabelas são populadas por regras que descrevem diferentes políticas de rede. Tais regras definem como os fluxos serão tratados pelo *switch*.

Na especificação da versão 1.3.1 do *openflow*, encontrada em FOUNDATION (2012), cada entrada na tabela de fluxo contém:

- ***Match Fields*:** campos de correspondência que serão comparados com o fluxo entrante no *switch*;
- ***Priority*:** prioridade corresponde à precedência da entrada do fluxo;
- ***Counters*:** contatadores que fornecem métricas atualizadas quando os pacotes são correspondidos;
- ***Instructions*:** instruções utilizadas para modificar o conjunto de ações ou o processamento do pipeline;
- ***Timeouts*:** quantidade máxima de tempo ou tempo ocioso antes que o fluxo expire pelo comutador;

- ***Cookie*:** valor de dados opaco escolhido pelo controlador. Pode ser usado pelo controlador para filtrar estatísticas, modificar e excluir o fluxo. Não é usado enquanto processa pacotes.

A partir da versão 1.3.0, o OpenFlow passou a suportar o controle das taxas de pacotes por meio de medidores de fluxos, introduzindo a tabela de medições (*Meter Table*). Uma tabela de medidores consiste em entradas que definem medidores por fluxo. Esses medidores permitem ao OpenFlow implementar várias operações simples de QoS, como limite de taxa de transferência ou ainda combinar com as filas das portas para implementar estruturas de QoS mais complexas como o DiffServ. Um medidor, diferente das filas, é anexado às entradas de fluxos de maneira exclusiva em uma tabela e vários medidores podem ser usados no mesmo conjunto de pacotes usando-os em tabelas de fluxo sucessivas (FOUNDATION, 2012).

Na especificação da versão 1.3.1 do *openflow*, encontrada em FOUNDATION (2012), cada entrada do medidor possui identificação única e contém:

- ***Meter identifier*:** um número inteiro de 32 bits para identificação exclusiva do medidor;
- ***Meter bands*:** uma lista não ordenada de bandas de medidores, em que cada banda especifica a taxa de banda e a maneira de processar o pacote;
- ***Counters*:** atualizado quando os pacotes são processados por um medidor.

Este Capítulo introduziu o paradigma SDN apresentando os seus principais conceitos e os principais componentes da sua arquitetura.

2.2 Qualidade de Serviço - QoS

Com o progresso da Internet, novos tipos de aplicativos e serviços de rede surgiram para os usuários finais. Essas novas demandas geram seus próprios fluxos específicos que precisam ser fornecidos pela Internet. No entanto, essas aplicações requerem diferentes tratamentos para seus próprios fluxos tornarem a entrega bem-sucedida. Por exemplo, alguns aplicativos exigem uma certa largura de banda, enquanto outros são mais sensíveis ao atraso. Atender esses requisitos exige um mecanismo(s) de qualidade de serviço (QoS) bem definido em uma rede. Os modelos de QoS atuais não são bem sucedidos o suficiente para resolver os problemas de qualidade de serviço dos paradigmas tradicionais de rede. Diante das limitações impostas pelas redes tradicionais, surge o paradigma SDN (Software Defined Networking), visto que uma de suas principais vantagens é a visão de rede global centralizada, a capacidade de programação e a separação do plano de dados e do plano de controle (KARAKUS; DURRESI, 2017).

Qualidade de Serviço é um termo amplo e em linhas gerais, pode alcançar várias definições. Para ISO(*International Organization for Standardization*), QoS é um conjunto de qualidades relacionadas ao comportamento coletivo de um ou mais objetos (ISO, 1998). Em

SABATA et al. (1997), a definição de QoS está relacionada a uma combinação de métricas e políticas, onde as métricas medem atributos quantificáveis específicos dos componentes e as políticas ditam o comportamento dos componentes do sistema. Já (TEITELBAUM; HANSS, 1998) diz que QoS é um termo frequentemente usado para se referir tanto ao desempenho de uma rede em relação às necessidades de aplicativos quanto ao conjunto de tecnologias que permitem que uma rede faça garantias de desempenho. Por fim, a CISCO (2009) diz que o objetivo da QoS é fornecer serviço de rede melhor e mais previsível, fornecendo largura de banda dedicada, *jitter* controlado e latência e perda de características melhoradas. QoS atinge esses objetivos, fornecendo ferramentas para gerenciar o congestionamento da rede, formação de rede tráfego, utilizando-se de maneira ampla área de links de forma mais eficiente e definindo políticas de tráfego em toda a rede. QoS oferece serviços de rede inteligente que, quando corretamente aplicados, ajudam a fornecer desempenho consistente e previsível.

2.2.1 Parâmetros de QoS

Oferecer QoS substancialmente significa proporcionar garantias de transmissão para certos fluxos de dados. A garantia de transmissão pode ser expressa como a combinação de alguns dos seguintes parâmetros (TEITELBAUM; HANSS, 1998):

Nível de Atraso (*latency*): É uma expressão utilizada para definir quanto tempo leva para um pacote de dados ir de um ponto designado para o outro.

Variação do atraso (*jitter*): É uma variação no atraso dos pacotes recebidos. No lado de envio, os pacotes são enviados em um fluxo contínuo com os pacotes espaçados com uma distância uniforme. Devido ao congestionamento da rede, ao enfileiramento impróprio ou aos erros de configuração, este fluxo constante pode tornar-se irregular ou o atraso entre cada pacote pode variar em vez de permanecer constante.

Taxa de transferência (*throughput*): Em termos gerais, é a taxa de entrega bem-sucedida de mensagens em um canal de comunicação.

Largura de banda (*bandwidth*): É a capacidade de transportar informações por meio de um canal de comunicação. Em resumo, determina a velocidade que os dados trafegam por meio de uma rede específica. Ou seja, quanto maior a largura de banda, maior será a velocidade da conexão, visto que por ela passará mais dados ao mesmo tempo.

Perda de pacotes: No processo de transmissão de unidades de dados (pacotes) entre um endereço de origem e um endereço de destino, um único ou vários pacotes podem ser perdidos na transferência e não conseguir chegar ao endereço de destino final. Isto é referido como perda de pacotes.

Confiabilidade: É uma propriedade dos sistemas de transmissão, podendo ser vista como a taxa de erros do meio físico. Na Internet, no entanto, protocolos como o TCP consideram que menos de 1% das perdas de pacotes tem causas físicas. O principal componente para expressar a

confiabilidade é, então, o roteamento, que pode atrasar os pacotes, alterar a sua ordem ou mesmo descartá-los quando as filas estão cheias (KAMIENSKI; SADOK, 2000).

2.2.2 Metodologias de QoS

A Internet está mudando muitos aspectos na vida das pessoas - negócios, entretenimento, educação e muito mais. As empresas usam a Internet e as tecnologias relacionadas à Web para ajudar a otimizar processos e desenvolver novos modelos de negócios. Por trás de todo esse sucesso está o tecido subjacente da Internet: o *Internet Protocol* (IP). Aplicativos diferentes têm necessidades variadas de atraso, variação de atraso (*jitter*), largura de banda, perda de pacotes e disponibilidade. Esses parâmetros formam a base da QoS. A rede IP deve ser projetada para fornecer a QoS necessária aos aplicativos.

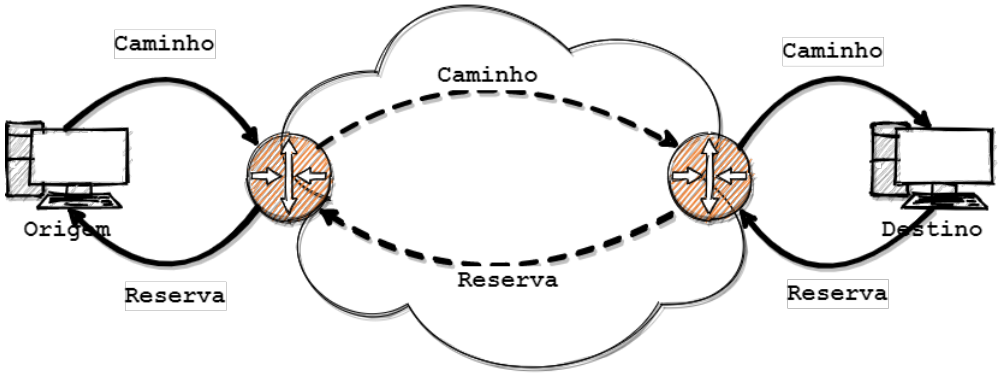
Para colaborar com a verdadeira QoS de ponta a ponta em uma rede IP, a IETF (*Internet Engineering Task Force*) definiu dois modelos de QoS que são muito bem aceitos atualmente: Serviços Integrados (IntServ) e Serviços Diferenciados (DiffServ) (CISCO, 2005).

O IntServ inclui dois tipos de serviço direcionados ao tráfego em tempo real: serviço garantido e preditivo. Ele integra esses serviços ao compartilhamento de link controlado e foi projetado para funcionar bem com multicast e unicast. Com os Serviços Integrados, os recursos (por exemplo, largura de banda) são gerenciados explicitamente para atender aos requisitos do aplicativo. Isso implica que a reserva de recursos e o controle de admissão são componentes fundamentais. Como a unidade de gerenciamento de recursos é o fluxo de dados do aplicativo, o gerenciamento de reserva de recursos por fluxo é necessário na rede. O *ReSource reserVation Protocol* (RSVP) é usado como protocolo de sinalização para reservas explícitas de recursos de ponta a ponta por fluxo de dados do aplicativo (PONNAPPAN et al., 2002).

O processo de reserva do protocolo RSVP ocorre através da propagação de mensagens RSVP salto a salto (*hop-by-hop*) na rede. A medida em que as mensagens são enviadas de um dispositivo para o próximo, o caminho entre a origem e o destino vai sendo delineado. Depois que o caminho é definido, outra mensagem é enviada no sentido inverso para estabelecer e manter a reserva dos recursos. Na circunstância de algum dos dispositivos não dispor dos recursos solicitados, as reservas já feitas até aquele ponto do caminho são desfeitas.

A figura 7 ilustra reservas de recursos utilizando IntServ. O path armazena informações sobre o caminho em cada um dos nós, os quais mantêm a caracterização do estado do caminho para o fluxo do transmissor. Após receber a mensagem path, o receptor envia uma mensagem de volta (RESV) ao transmissor, pelo mesmo caminho que a mensagem path foi enviada ao receptor. Como o receptor é responsável por requisitar o QoS, a mensagem especifica a qualidade de serviço pretendida e estabelece o estado de reserva em cada nó no caminho. Após receber com sucesso a mensagem *reservation-request*, o transmissor inicia o envio de dados. Tanto o transmissor quanto o receptor continuam enviando essas mensagens de tempos em tempos, a fim

Figura 7 – Reserva de Recurso IntServ



Fonte: Adaptado (MENDONCA, 2014)

de manter o estado da sessão, caso contrário, a reserva é cancelada (PONNAPPAN et al., 2002; MENDONCA, 2014).

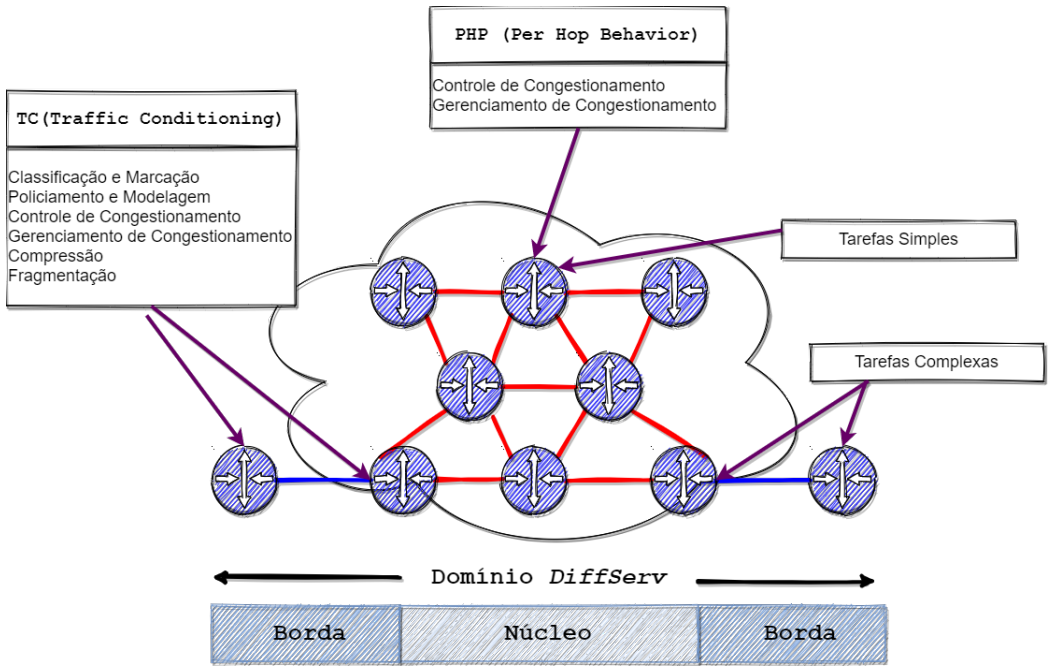
O DiffServ surgiu como uma abordagem de Classe de Serviço, *Class of Service* (CoS), para fornecer QoS soberano ao "melhor esforço", método padrão de entrega de dados em redes IPs. Os Serviços Diferenciados buscam um mecanismo relativamente simples e leve que não dependa inteiramente da reserva explícita de recursos por fluxo, em contraste aos Serviços Integrados, que usam uma abordagem mais rigorosa e complexa à QoS. Com os Serviços diferenciados, não há necessidade de sinalizar os parâmetros que descrevem o tipo de serviço desejado ou de fazer reservas de recursos explicitamente. Ao invés disso, os parâmetros são programados em roteadores e uma classe de serviço é selecionada mediante assinatura de um SLA (*Service Level Agreement*) ou pela marcação apropriada de pacotes IP através do padrão de 6 bits chamado DSCP (*Differentiated Services Code Point*) (PONNAPPAN et al., 2002).

De acordo com a figura 8, os dispositivos localizados nas extremidades da rede (dispositivos de acesso e borda) normalmente são responsáveis pela classificação e marcação do tráfego. Isso significa que são eles quem dividem o tráfego em classes e marcam cada pacote, para que estes pacotes não precisem ser novamente classificados quando alcançarem os dispositivos localizados no interior do backbone.

Esta é uma tarefa de alta complexidade, já que analisa vários campos do pacote para identificar a classe a qual ele pertence. Além das funções de classificação e marcação, estes dispositivos também executam outras funções como policiamento e moldagem, gerenciamento de filas, compressão e fragmentação. O conjunto de funções desempenhadas por estes roteadores é denominado Traffic Conditioning (TC).

Já os dispositivos do interior do domínio DiffServ têm a função de tratar os pacotes que foram previamente marcados e classificados. A tarefa principal destes dispositivos é priorizar o tráfego segundo os recursos alocados para cada classe. Esta tarefa é considerada de menor

Figura 8 – Domínio DiffServ



Fonte: Adaptado (MENDONCA, 2014)

complexidade e também de menor custo de processamento para o roteador. O conjunto de funções desempenhadas pelos roteadores localizados no interior da rede é denominado *Per Hop Behavior* (PHB) (MENDONCA, 2014).

2.2.3 Relação entre QoS e SDN

Na sessão 2.1 foram discutidas as principais e mais importantes características de uma SDN. Tendo em vista esses conceitos, KARAKUS e DURRESI (2017) afirmam que a QoS pode se beneficiar das vantagens da SDN, por exemplo: o encaminhamento baseado em fluxo permite que as redes direcionem diferentes fluxos de aplicativos em diferentes tratamentos. A atualização de regras de fluxos dinâmicos permite que os operadores de rede atualizem as regras instaladas nos dispositivos de rede imediatamente, sem interromper as operações do dispositivo. O SDN também possibilita a análise de fluxo/pacote para adquirir os campos de cabeçalho. Com uma visão global da rede, é possível manter estados relacionados para um caminho completo de um fluxo. Além disso, é possível monitorar as estatísticas da rede baseadas em diferentes níveis, como fluxo, porta, dispositivo e assim por diante. Para mais, as redes SDN que utilizam o protocolo OpenFlow, possibilitam o gerenciamento de filas e operações de agendamento com protocolos OF-CONFIG e OVSDB, por exemplo.

Por fim, a SDN pode ser utilizada para fornecer QoS de várias maneiras, como provisionamento baseado em virtualização, gerenciamento de políticas e mecanismos de entrega de conteúdo devido a alguns de seus recursos, como o conceito de controle por fluxo e o roteamento

baseado em vários campos de cabeçalhos(KARAKUS; DURRESI, 2017).

2.3 *Big Data*

No início da era da informação, os grandes volumes de dados pareciam descrever adequadamente as mudanças tecnológicas, culturais e econômicas do início dos anos 2000. A partir de então, foi possível ter acesso a várias novas formas de dados: dados da Web, dados de dispositivos móveis e, mais recentemente, dados de redes de sensores. Desde então, uma verdadeira avalanche de informações surge a todo momento. Para ir direto ao ponto e explicar porque a explosão de dados digitais tem se tornado tão importante, a frase atribuída a W. Edwards Deming e PeterDrucker, ganha destaque, ela diz: "Você não pode gerenciar o que não mede". Partindo desse princípio, quando um grande volume de dados oriundos de várias origens (Sistemas, sensores, planilhas, etc) são medidos e analisados em conjunto, informações valiosas para o negócio tendem a surgir. Traduzir esse conhecimento em uma decisão melhorada no processo de fabricação ou no desempenho de uma tarefa é uma realidade possível a partir da análise de *big data* (MCAFEE et al., 2012).

Essa sessão apresenta os conceitos e definições de *big data*, as fases, aplicações e alguns desafios.

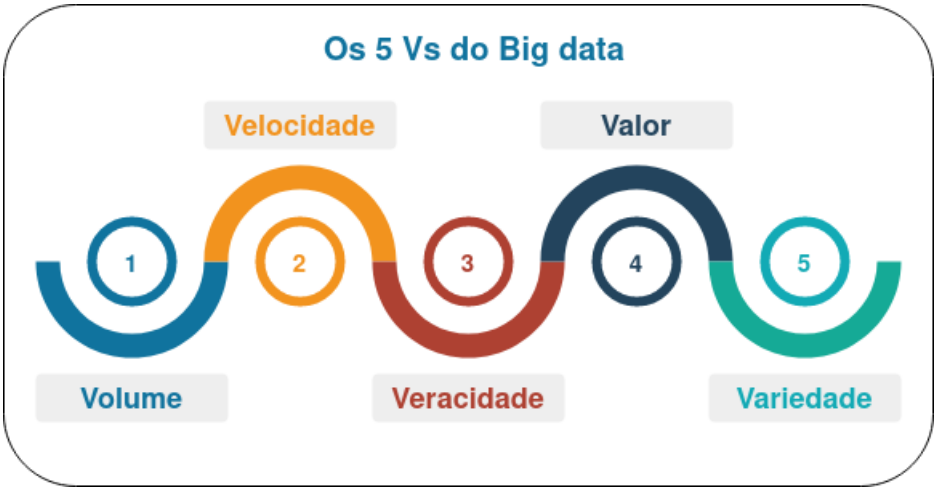
2.3.1 Definição

Big data é um conceito relativamente novo e abstrato, possui, entre outras características, a ideia de dados massivos. No entanto, está longe de ser um assunto tratado de forma tão simplória. Existem muitas definições na literatura para o termo *big data*. Provost e Fawcett (2013), Hu et al. (2013) e Fisher et al. (2012) conceituam *big data* como sendo um conjunto de dados grandes demais para os sistemas tradicionais de processamento de dados e, portanto, requerem novas tecnologias. Já para Jin et al. (2015), o conceito é tratado por meio de uma perspectiva macro e afirma que pode ser considerado um vínculo que sutilmente conecta e integra o mundo físico, a sociedade humana e o ciberespaço. Para este conceito, considera-se que o mundo físico reflete o ciberespaço, incorporado como *big data*, através da Internet, da Internet das Coisas e de outras tecnologias da informação. O livro de Mayer-Schönberger e Cukier (2013) define o termo como sendo a capacidade da sociedade de obter informações de novas maneiras para produzir informações úteis ou bens e serviços de valor significativo. Além desses, o estudo de Manyika (2011) classifica como um conjuntos de dados cujo tamanho está além da capacidade das ferramentas de software de banco de dados típicas de capturar, armazenar, gerenciar e analisar.

Apesar das definições serem tratadas de forma distinta por alguns autores, é possível observar algumas características comuns quando falam de *big data*. Algumas dessas características foram articuladas por Laney (2001) e ficaram conhecidas como *mainstream* de *big data* em 3Vs.

Os 3Vs representam os termos **V**olume, **V**elocidade e **V**ariedade. No entanto, estudos decorrentes (CUI; YU; YAN, 2016; JIN et al., 2015) apontaram que a definição de 3Vs é insuficiente para explicar os grandes dados atuais. Assim, as dimensões **V**eracidade, um termo cunhado pela IBM e **V**alor, introduzido pela Oracle, agregaram positivamente para tornar o tema mais abrangente (KAUR; SOOD, 2017). Desta feita, o *big data* pode ser descrito pelas seguintes características ilustradas na figura 9:

Figura 9 – Os 5Vs do *Big Data*



Fonte: Autor (2020)

O Volume é a característica que apresenta o desafio mais imediato às estruturas de TI convencionais. Refere-se a grande quantidade de dados que estão sendo gerados, coletados e processados. Por exemplo, uma organização pode coletar dados de fontes variadas, incluindo transações financeiras, mídias sociais e informações de sensores ou dados transmitidos de máquina para máquina. Atualmente, um bom exemplo para se pensar em grandes volumes de dados seria o Facebook, que rotineiramente lida com milhões de postagens e imagens ou o Google, que recebe mais de 3,5 bilhões de pesquisas por dia (STATS, 2019). Além disso, existem os milhões de registros de dados que são coletados a partir de tecnologias de sensores associadas a transporte, clima, sistemas ambientais e assim por diante. O benefício obtido com a capacidade de processar grandes quantidades de informações é a principal atração da análise de *big data* (ANURADHA et al., 2015).

A Velocidade é um fator muito importante para lidar com *big data*. Essa dimensão está relacionada à velocidade crescente na qual os dados são criados, processados e movidos entre diferentes sistemas e dispositivos (YOUNAS, 2019). Citando alguns números é possível ter uma noção sobre como tudo isso acontece atualmente e qual a velocidade. Hoje, cerca de 40% da população mundial está conectada a Internet. Em 1995, esse número era de apenas 1%. Existem mais de 1,5 bilhão de sites na Internet. São aproximadamente 22 bilhões de fotos enviadas ao Instagram de Janeiro a Outubro de 2019. Em um único dia, o YouTube tem mais de 6 bilhões de visualizações (STATS, 2019).

A Veracidade refere-se à qualidade dos dados, como correção, consistência, confiança, segurança e confiabilidade. Quando se lida com um alto volume, velocidade e variedade de dados, não é possível que estejam 100% corretos, haverá dados sujos. A qualidade dos dados capturados pode variar muito e a precisão dos dados analisados depende da veracidade dos dados de origem. De forma geral, a veracidade garante que os dados utilizados sejam confiáveis, autênticos e esteja protegidos contra acessos não autorizados (YOUNAS, 2019).

O Valor refere-se ao processo de extrair conhecimento de *big data*. Isso inclui diferentes tipos de benefícios que podem ser derivados do processamento e análise dos dados. Os exemplos incluem valor monetário, valor social, valor de pesquisa/educação e assim por diante (KAUR; SOOD, 2017).

A Variedade refere-se à heterogeneidade dos tipos de dados. Os dados de texto, imagem, áudio e vídeo capturados por sensores, atuadores, transações comerciais, redes sociais e ambientes inteligentes contribuem para a variedade de dados. A maioria desses dados está em formato não estruturado. Em suma, são os diferentes tipos de dados que podem ser usados (juntos) para alcançar as informações ou resultados desejados (KAUR; SOOD, 2017).

2.3.2 Cadeia de Valor do *Big Data*

Um sistema de *big data* é complexo e fornece funções para lidar com diferentes fases do ciclo de vida dos dados digitais, desde o nascimento até a destruição. Para obter *insights* e valor comercial, é preciso fazer investimentos em alguns elementos-chave, dentre os quais destacam-se quatro: aquisição/geração, armazenamento de dados, análise de dados e visualização/saída de dados (MARR, 2019; HU et al., 2014). A figura 10 representa bem estas etapas.

Figura 10 – Cadeia de Valor do *Big Data*

Fonte: Adaptado (HU et al., 2014)

É através da **aquisição** que os dados são apresentados. Esse componente se comporta como captador de dados de várias fontes, desde os registros de vendas, banco de dados de clientes, *feedback*, canais de mídias sociais, listas de marketing, arquivos de e-mails e quaisquer dados obtidos a partir do monitoramento ou medições do ponto de vista de operações. Em alguns casos é necessário fazer investimentos em infraestrutura para obter novos dados. Os requisitos de infraestrutura para captura de dados dependem do tipo de dados necessários, mas as principais opções podem incluir: sensores (em máquinas, edifícios ou veículos, embalagens,

etc), aplicativos que geram dados do usuário, vídeo CFTV (Circuito Fechado de TV), alterações no site que solicitam aos clientes mais informações e perfis de mídias sociais (MARR, 2019).

A segunda fase diz respeito ao **armazenamento** e gerenciamento persistente de conjuntos de dados em larga escala. Um sistema de armazenamento de dados pode ser dividido em duas partes: infraestrutura de hardware e gerenciamento de dados. A infraestrutura de hardware consiste em um conjunto de recursos compartilhados de TIC, organizados de maneira elástica para várias tarefas em resposta à demanda instantânea. A infra-estrutura de hardware deve poder ser ampliada e ampliada ser reconfigurada dinamicamente para lidar com diferentes tipos de ambientes de aplicativos. O software de gerenciamento de dados é implantado na parte superior da infraestrutura de hardware para manter conjuntos de dados em larga escala. Além disso, para analisar ou interagir com os dados armazenados, os sistemas de armazenamento devem fornecer várias funções de interface, consultas rápidas e outros modelos de programação (HU et al., 2014).

Para descobrir algo útil usando dados, é necessário processá-los e **analisá-los**. A análise de dados utiliza métodos ou ferramentas analíticas para inspecionar, transformar e modelar dados com objetivo de extrair valor, sugerir conclusões e apoiar decisões. Para isso, é necessário preparar os dados, construir modelos analíticos e só então, tirar conclusões dos *insights* obtidos. Atualmente, existem softwares específicos para esse fim e empresas como IBM, Oracle e Google já comercializam esse tipo de solução (MARR, 2019; HU et al., 2014).

A **visualização ou saída de dados** é o termo usualmente utilizado para representar as ideias coletadas na análise de dados para os tomadores de decisões. Esses resultados podem assumir a forma de breves relatórios, gráficos, figuras e recomendações importantes. Gradualmente as empresas estão abandonando os relatórios gigantescos com várias páginas e gráficos difíceis de serem compreendidos. É claramente irreal esperar que pessoas ocupadas vasculhem montanhas de dados com infinitos apêndices de planilhas e extraiam as principais mensagens (MARR, 2019; HU et al., 2014).

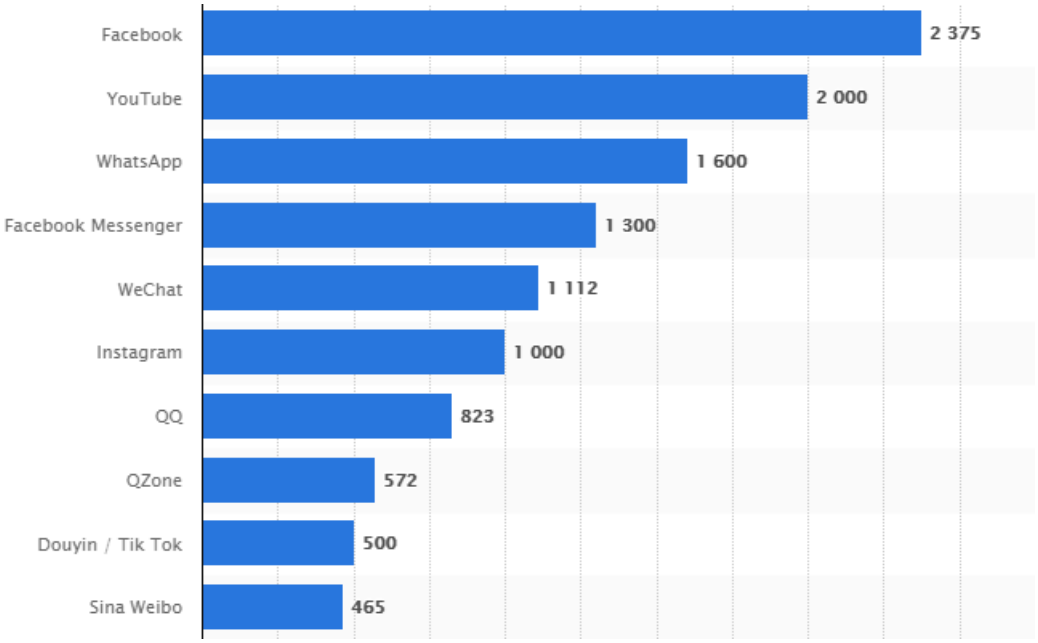
2.3.3 Aplicações de *Big Data*

São grandes as possibilidades de aplicação de *big data*. Diversos segmentos de mercado como mídias sociais, aplicativos móveis, Internet das coisas, varejo, indústria, saúde, mercado financeiro e tecnológico, entre outros, são grandes geradores de fluxos e portanto propensos a utilizarem a tecnologia.

A utilização de mídias sociais vem crescendo muito e isso pode ser percebido pelo grande número de usuários ativos, ilustrados na figura 11. De conhecimento desses números, dá para imaginar a quantidade de dados gerados por eles a todo momento. Boa parte desses dados são gerados através das mídias sociais mais populares, dentre elas estão o Facebook, Youtube, Whatsapp, Twitter e Instagram (KAUR; SOOD, 2017). A abertura para que todos ouçam, expressem suas opiniões e construam novos relacionamentos abriu caminho para a

criação de riqueza de dados. Nesse embalo, até as grandes corporações começaram a usar as mídias sociais como canal de negócio e isso tem chamado a atenção de cientistas de dados para explorar o uso das redes sociais em diversas áreas do conhecimento (MOHANTY; BHUYAN; CHENTHATI, 2015).

Figura 11 – Usuários Ativos em Redes Sociais (Milhões) - Julho de 2019



Fonte: (STATISTA, 2019)

As principais estratégias ao analisar mídias sociais incluem marketing, promoção da marca, identificação de novas oportunidades de vendas, atendimento ao cliente, previsão de eventos futuros, promover novos negócios, etc.

No setor da saúde o *big data* está ganhando importância devido às características do negócio envolvendo um enorme conjunto de dados de clientes como registros médicos eletrônicos, registros laboratoriais, correspondências médicas, reclamações, entre outros. As análises avançadas desses dados são usadas para melhorar o atendimento e os resultados dos clientes, aumentar a eficiência e manter os custos mínimos. Além disso, é possível que se façam investigações detalhadas para detectar efeitos colaterais adversos dos medicamentos e assim fazer correções rápidas nos medicamentos. Esses são apenas alguns exemplos entre várias possibilidades (MOHANTY; BHUYAN; CHENTHATI, 2015).

Já nos grandes varejos, é possível utilizar o *big data* em várias operações, incluindo gerenciamento de inventário, recomendações de produtos, rastreamento de dados demográficos de clientes e gerenciamento dos efeitos de *recalls* de produtos. Deste modo, é possível usar esses dados para melhorar a qualidade do serviço e aumentar a fidelidade do cliente (MOHANTY; BHUYAN; CHENTHATI, 2015).

Os serviços de telecomunicações crescem em todo o mundo e estão entrando em vários

segmentos de mercado com serviços de voz, vídeo, dados, etc. Estruturas de análise de *big data* podem ser utilizadas como base para formulação de estratégias de melhorias no negócio, por exemplo: detecção de padrões de utilização, registros de atendimentos, comentários em redes sociais, entre outros (MOHANTY; BHUYAN; CHENTHATI, 2015).

Em pontos mais específicos, como por exemplo nas redes definidas por software, o *big data* pode beneficiar temas importantes como a engenharia de tráfego, o design de camada cruzada e também pode ajudar a derrotar ataques de segurança, entre outras possibilidades (CUI; YU; YAN, 2016).

2.3.4 Desafios de *Big Data*

O *big data* está se tornando mais popular entre as empresas em todos os setores. No entanto, colocar em prática um projeto de *big data* não é fácil e exige a superação de numerosos desafios (BROWN, 2019). De acordo com Bean (2017), um estudo realizado em 2017 com executivos da **Fortune 1000** mostrou que 95% dos empreendedores pesquisados realizaram um projeto de *big data* nos últimos cinco anos, mas apenas 48,4% dos executivos corporativos pesquisados indicaram que sua empresa alcançou "resultados mensuráveis"com seus investimentos.

Gerenciar o crescimento de dados é, sem dúvidas, um dos maiores desafios de *big data* a serem superados. Para vencer esse desafio, as empresas podem usar diferentes tecnologias emergentes como ferramentas NoSQL, Hadoop, Spark e outros softwares analíticos, bem como software de *business intelligence*, intelegência artificial e aprendizado de máquina para obter as informações de que precisam.

Gerar *insights* rapidamente é um desafio bastante interessante. As empresas não querem apenas armazenar os dados que geram. Elas estão interessadas em atingir objetivos claros com o uso da tecnologia como redução de despesas, a implementação de uma cultura baseada em dados, inovação, a aceleração da implantação de novas capacidades e serviços e o lançamento de novos produtos e serviços. Para ajudá-los a atingir essa velocidade, podem usar uma nova geração de ferramentas analíticas que reduzem significativamente o tempo necessário para gerar relatórios (BROWN, 2019).

Além dos desafios já citados, existem outros, a exemplo: integrar diversas fontes de dados (aplicativos de negócios, redes sociais, e-mails, documentos de funcionários, etc), encontrar profissionais qualificados, validar os dados, alcançar segurança dos dados, resistência organizacional, escalabilidade, gerenciamento de energia (BROWN, 2019; HU et al., 2014).

2.4 Coleta de Tráfego

Coletar informações dos fluxos que trafegam na rede é uma das etapas iniciais para identificação, classificação, caracterização ou qualquer outra atividade cujo o interesse seja

de analisar o comportamento do tráfego de uma rede de computadores. Uma das questões de pesquisa da revisão sistemática abordada no Capítulo 3 deste trabalho, indaga como a tarefa de extração é realizada em redes SDN.

No cenário apresentado pela revisão sistemática, a maioria dos autores (JOSE; NAIR; PAUL, 2017), (SEMENCIUC et al., 2016), (KOLOMVATSOS et al., 2017), (ALWASEL et al., 2017) e (DESAI; S; T, 2015) relataram que desenvolveram algum software para realizar essa tarefa em SDN. O desenvolvimento de aplicativos para esse fim tem se tornado comum graças às interfaces *southbound* que possibilitam a coleta de estatísticas dos fluxos e que permite, inclusive, criação de softwares de monitoramento da rede, a exemplo do software de monitoramento proposto por Silva et al. (2016). No entanto, existem outros meios de coleta descritos na literatura. Os trabalhos (BAKHSHI; GHITA, 2015) e (JIANG et al., 2010), por exemplo, utilizam logs do NetFlow (CISCO, 2012). Em (DESAI; S, 2015) e (DESAI; NAGEGOWDA; NINIKRISHNA, 2016), softwares *sniffers* e em (JAIN et al., 2016), o protocolo SNMP(*Simple Network Management Protocol*).

O protocolo OpenFlow, que é um dos mais populares e mais utilizados em Redes definidas por Software, possui contadores nos níveis de tabelas de fluxos, entrada de fluxo, porta, fila, grupo, grupo balde, medidor e faixa do medidor, que são capazes de capturar as estatísticas dos fluxos nos comutadores de uma SDN. As informações de um contador podem ser acessadas através das mensagens OpenFlow, como são conhecidas. As mensagens são divididas em três tipos: controlador para comutador, assíncrona e simétrica (FOUNDATION, 2012). Os detalhes dos tipos de mensagens OpenFlow são discutidos na subseção 2.1.8. Dessa forma, a coleta de estatísticas de fluxo dos comutadores em uma SDN pode ser realizada por meio de desenvolvimento de simples aplicativos na camada de aplicação da SDN ou por meio de módulos no próprio controlador (se possuir código aberto).

Os contadores são definidos pela especificação dos fluxos, por exemplo, se a coleta for realizada por grupo, apenas as informações referentes a contagem de pacotes e contagem de bytes poderão ser obtidas. No entanto, se a coleta for realizada por fila, informações como pacotes transmitidos, bytes transmitidos, erros, duração em segundos e nanosegundos poderão ser colhidas. Os detalhes de cada especificação dos contadores disponível no protocolo OpenFlow versão 1.3, podem ser conferidos na tabela 2.

Tabela 2 – Medidores do OpenFlow na Versão 1.3

Especificação	**Contador**
Por tabela de fluxo	Contagem de referência (entradas ativas) Pesquisas de Pacotes Correspondências de Pacotes
Por entrada de fluxo	Pacotes recebidos Bytes recebidos Duração (segundos) Duração (nanossegundos)
Por porta	Pacotes recebidos Pacotes transmitidos Bytes recebidos Bytes transmitidos Drops recebidos Drops transmitidos Erros recebidos Erros de transmissão Erros de alinhamento de quadros recebidos Erros de saturação recebidos Erros de CRC recebidos Colisões Duração (segundos) Duração (nanossegundos)
Por fila	Pacotes transmitidos Bytes transmitidos Erros de saturação transmitidos Duração (segundos) Duração (nanossegundos)
Por grupo	Contagem de referência (entradas de fluxo) Contagem de pacotes Contagem de bytes Duração (segundos) Duração (nanossegundos)
Por grupo balde	Contagem de pacotes Contagem de bytes
Por medidor	Contagem de fluxo Contagem de pacotes de entrada Contagem de bytes de entrada Duração (segundos) Duração (nanossegundos)
Por faixa de medidor	Contagem de pacotes na banda Contagem de bytes em banda

Fonte: (FOUNDATION, 2012)

2.5 Caracterização de Tráfego

Com o grande crescimento das redes de computadores, em particular da Internet, tornou-se de grande importância entender o comportamento do tráfego de rede para o gerenciamento adequado e eficiente dos recursos. O trabalho de Claffy (1995) já chamava atenção para essa necessidade que é um assunto notório até os tempos atuais. Em seu trabalho, Claffy (1995) utilizou medições no Backbone da NSFNET(*National Science Foundation Network*) para coletar e analisar o comportamento de algumas métricas, como o crescimento do tráfego ao longo do tempo, aplicações que eram responsáveis pela maior porção do tráfego, número de endereços IP utilizados ao longo do tempo e os usuários responsáveis pela maior utilização dos enlaces. São informações importantes e que ajudam os administradores a obter uma visão realista do uso de recursos e os norteiam para posteriormente aplicar um gerenciamento de recursos mais eficiente.

Diversas técnicas têm sido utilizadas para compreender o comportamento do tráfego em redes de computadores, entre elas, técnicas de *cluster* não supervisionados que particionam fluxos IP em *clusters* com base em diferentes recursos de tráfego de interesse, como número de fluxos, sessões, uso de aplicativos e hosts. Apesar da popularização dessa técnica em redes tradicionais, pouco tem se falado na compreensão dos padrões de tráfego no contexto de SDN, usando especificamente as primitivas de contabilidade de fluxo fornecidas pelo protocolo OpenFlow, vistos na tabela 2, para esse fim (BAKHSHI, 2017).

Alguns trabalhos na literatura citam a abordagem da caracterização por meio de técnicas de *cluster* não supervisionados. Os trabalhos de Jiang et al. (2010) e BAKHSHI e GHITA (2015) são exemplos claros da utilização dessa técnica. Eles empregam o algoritmo de *cluster* K-Means para traçar perfis de tráfego, um com base em dados agregados a nível de prefixo de rede e outro a nível de usuário usando IP de origem. É importante destacar que nenhum utiliza as primitivas de contabilidade de fluxo fornecidas pelo protocolo OpenFlow, embora o trabalho de BAKHSHI e GHITA (2015) tenha pretensões de gerenciamento de SDN.

O artigo de Jiang et al. (2010) trabalha com caracterização no nível de prefixo de rede. Para isso, foca na criação de perfil de tráfego agregado no nível de prefixo da rede utilizando o algoritmo de *cluster* K-Means, com intuito de descrever padrões estruturais nos dados de medição da rede, bem como obter informações a partir deles. O trabalho constrói perfis comportamentais por meio de recursos de interesse que são pertinentes ao gerenciamento de tráfego, por exemplo, volume de tráfego agregado diário, distribuição de tráfego no tempo, distribuição de tráfego no espaço , distribuição de tráfego na aplicação, distribuição do tamanho do fluxo e balanço de tráfego na direção do fluxo.

O trabalho de BAKHSHI e GHITA (2015) busca traçar padrões estatísticos significativos de tráfego ou extrair perfis de comportamento do usuário através da utilização de aplicativos mapeados previamente e a partir dessas informações sugerir o uso em vários caminhos, que variam da segurança da rede à análise de tendências on-line. Para isso, o estudo utilizou registros

de fluxo coletados de uma rede residencial com 250 usuários que foram agrupados em nível de aplicativos. Através do algoritmo de agrupamento K-Means, foi possível traçar quatro perfis de usuários na rede e a partir daí conseguiu-se identificar, por exemplo, a distribuição de tráfego do aplicativo entre os perfis do usuário, o volume total de tráfego, a soma dos bytes de entrada e saída por dia para cada perfil de tráfego, a quantidade de usuário por perfil.

Em (SILVA et al., 2015), pode-se observar algo mais especifico. O objetivo do trabalho é a caracterização e modelagem do tráfego de imagens médicas em redes de computadores hospitalares. O modelo resultante do trabalho pode ser utilizado como entrada em experimentos de avaliação ou planejamento de capacidade como a ampliação da infraestrutura da rede voltada para a saúde. O trabalho conseguiu demostrar através dos dados reais de um hospital de pequeno porte que o tráfego de rede existente é composto, em sua maioria, por tráfego do protocolo TCP, cuja participação corresponde a mais de 90% do volume de dados coletados. Outro ponto importante foi o índice de participação do protocolo DICOM (*Digital Imaging and Communications in Medicine*) com 20%, atingindo esse valor com uma baixa quantidade de pacotes e permitindo a identificação desse tráfego como elefante.

É possível notar que a caracterização de tráfego é uma questão ampla e complexa, que envolve diferentes variáveis e métricas, dependendo do que se deseja estudar. Nesse sentido, a caracterização escalável e adequada do tráfego de rede em ambientes SDN é, portanto, um requisito significativo para implementar e otimizar políticas de gerenciamento e reforçar a segurança (BAKHSHI, 2017; VILELA, 2006).

2.6 Classificação de Tráfego

Na última década, a classificação de tráfego de rede sofreu uma inovação significativa e agora é um pilar indispensável da funcionalidade de redes de computadores (HAYES et al., 2018). A quantidade de dados que flui pelas redes de telecomunicações e a complexidade dos aplicativos que os geram continuam aumentando e isso deu sentido à classificação de tráfego em uma variedade de atividades relacionadas à rede, incluindo anomalia de rede, detecção de violação de intrusão e segurança, engenharia e gerenciamento de tráfego, provisionamento de qualidade de serviço, contabilidade, solução de problemas e avaliação de conformidade de contratos de nível de serviço (HAYES et al., 2018; AMARAL et al., 2016).

Uma abordagem promissora para explorar a escala e as complexidades da classificação de tráfego é a rede definida por software. A arquitetura SDN abre novos caminhos nessa área e fornece um ambiente ideal para inovações em redes por seu dinamismo, flexibilidade, adaptabilidade, simplicidade e alta capacidade de gerenciamento. Segundo Fan e Liu (2017), existem três categorias de métodos importantes para classificação de tráfego: baseado em porta, carga útil e aprendizado de máquina. As subseções seguintes detalham as categorias.

2.6.1 Classificação Baseada em Porta e em Carga Útil

A classificação baseada em porta é uma das técnicas mais antigas e bem-sucedida, até certo ponto, porque muitos aplicativos usam números de porta fixos atribuídos pela IANA (*Internet Assigned Numbers Authority*). No entanto, nos últimos tempos, algumas limitações a essa abordagem se tornaram óbvias. Apareceu um grande número de aplicativos que não possuem números de porta registrados e muitos deles usam mecanismos dinâmicos de negociação de portas para se esconder de *firewalls* e ferramentas de segurança de rede. Além disso, o uso de criptografia, ofuscação e proxy da camada IP também pode ocultar o cabeçalho TCP ou UDP, impossibilitando a descoberta dos números de porta originais (AMARAL et al., 2016).

A abordagem de carga útil, também conhecida como *Deep Packet Inspection* (DPI), é atualmente uma das técnicas mais utilizadas. Os sistemas de DPI inspecionam a carga útil dos pacotes, procurando padrões que identificam o aplicativo. Esses padrões, também chamados de assinaturas, são definidos a partir de expressões regulares avaliadas por autômatos que funcionam de maneira sequencial, exigindo uma grande quantidade de memória. Isso pode causar um problema de escalabilidade ainda mais grave, pois o DPI é executado no caminho da comunicação. Além disso, as leis de privacidade e a criptografia podem levar a uma carga útil de tráfego inacessível (AMARAL et al., 2016; FAN; LIU, 2017).

2.6.2 Classificação Baseada em Aprendizado de Máquina

As técnicas de Aprendizado de Máquina, ou *Machine Learning* (ML), também podem ser usadas para classificação de tráfego. Métodos baseados em aprendizado de máquina podem superar algumas das limitações de abordagens baseadas em porta e carga útil, pois utilizam estatísticas de tráfego independentes do protocolo do aplicativo, como duração do fluxo, variação do comprimento do pacote, tamanho máximo ou mínimo do segmento, tamanho da janela, tempo de ida e volta e tempo de chegada de pacotes. Além disso podem levar a custos computacionais mais baixos e identificar o tráfego criptografado facilmente (AMARAL et al., 2016; FAN; LIU, 2017).

A classificação de tráfego pode utilizar métodos baseado em aprendizado de máquina supervisionado ou não supervisionado. No aprendizado supervisionado, é necessário obter conjuntos de dados de treinamento rotulados, algo que pode ser desafiador em redes de computadores devido à dificuldade em obter amostras de fluxo de rede anotadas com precisão em uma ampla gama de aplicativos e na taxa em que novos aplicativos podem surgir (AMARAL et al., 2016; FAN; LIU, 2017).

No aprendizado não supervisionado, os dados fornecidos não são etiquetados, normalmente é uma técnica usada para tarefas de *cluster*, onde os algoritmos agrupam os dados em diferentes *clusters* de acordo com as semelhanças nos valores dos recursos. O objetivo dessa técnica é encontrar relacionamentos desconhecidos nos dados, encontrando padrões de similaridade

entre as várias observações (AMARAL et al., 2016; FAN; LIU, 2017)

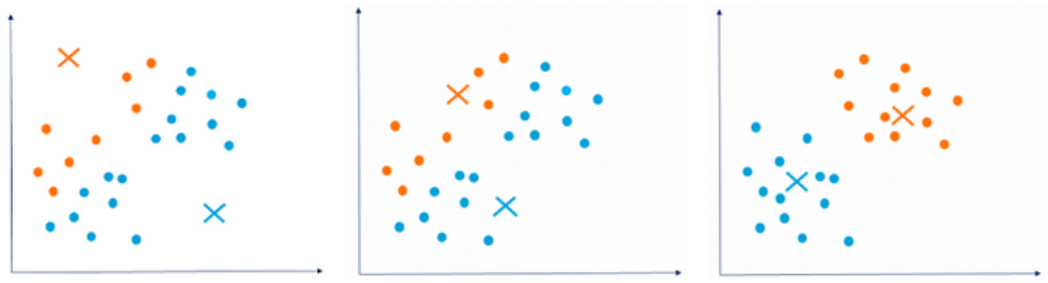
Esse trabalho utiliza um dos algoritmos mais populares de aprendizado de máquina não supervisionado, o K-Means. O método k-means é uma técnica de agrupamento amplamente usada e estudada que busca minimizar a distância quadrática média entre pontos no mesmo agrupamento. Funciona da seguinte forma: dado um conjunto n de pontos reais em d - *dimensões* em um espaço, R^d, e um número inteiro k, onde k é o número de centros ou centroides em R^d (KANUNGO et al., 2002; FRäNTI; SIERANOJA, 2019). O algoritmo começa selecionando k pontos de dados aleatórios como o conjunto inicial de centróides, que é então aprimorado por duas etapas subsequentes. Na primeira etapa, a de atribuição, cada ponto é colocado no *cluster* do centroide mais próximo. Na etapa seguinte, de atualização, o centroide de cada *cluster* é recalculado como a média de todos os pontos de dados atribuídos ao *cluster*. Juntas, essas duas etapas constituem uma iteração de k-means. O algoritmo é iterado um número fixo de vezes ou até a convergência, quando nenhuma melhoria adicional é obtida (FRäNTI; SIERANOJA, 2019).

Algumas métricas são utilizadas para calcular a distância entre os pontos e os centros do *cluster*. Uma das mais utilizadas é a métrica Euclidiana, mas existem outras, como Manhattan e Minkowski, por exemplo. A distância Euclidiana é simplesmente a distância geométrica no espaço multidimensional. Ela é calculada como:

$$dist(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$

A figura 12 mostra uma sequência de três gráficos que representam bem as interações do k-means e a movimentação dos centroides, a medida em que o centro de cada *cluster* é recalculado. No primeiro gráfico, dois pontos iniciais aleatórios, representados pela letra X, são inseridos; no gráfico seguinte o centroide de cada *cluster* é calculado com a média de todos os pontos; o ultimo gráfico mostra a convergência alcançada.

Figura 12 – Movimentação dos Centroides no k-means



Fonte: (SANTANA, 2019)

Em SDN, o k-means pode ser utilizado par alcançar vários objetivos, dentre eles, a exploração e detecção automática de padrões complexos nos fluxos de tráfego que possuam comportamento semelhante. Entender o comportamento da rede é uma etapa fundamental para uma gestão eficiente dos recursos da rede.

3

Revisão Sistemática

Este capítulo é totalmente dedicado a discutir sobre a revisão sistemática da literatura realizada para nortear essa pesquisa. A revisão sistemática é um tipo de estudo secundário e, para Kitchenham e Charters (2007), esse tipo de estudo é uma forma de responder questões de pesquisa relevantes para uma área de interesse por meio de identificação, avaliação e interpretação de estudos primários. Dessa forma, nas próximas sessões, será possível apreciar o protocolo da revisão sistemática e suas etapas, a exemplo da definição das questões de pesquisa, estratégias de busca e critérios de seleção, bem como a análise dos resultados e a conclusão.

3.1 Protocolo da Revisão Sistemática

A revisão sistemática conduzida neste estudo seguiu as diretrizes propostas por Kitchenham e Charters (2007). Esse tipo de estudo é uma forma de identificar, avaliar e interpretar todas as pesquisas disponíveis e que são relevantes para uma questão de pesquisa particular, área temática ou de interesse.

As subseções a seguir descrevem a elaboração do protocolo utilizado para confecção desse estudo.

3.1.1 Definição das Questões de Pesquisa

Para este trabalho, uma questão de pesquisa principal **(QP)** foi definida: "Como o *big data* tem sido explorado para auxiliar o QoS em Redes Definidas por Software?". As 5 (cinco) questões complementares são derivadas da questão principal. Ao responder as questões de pesquisa, espera-se obter um estudo rico sobre o tema abordado. As questões de pesquisa foram definidas da seguinte forma:

QP1. Quais são os principais objetivos e focos de pesquisa sobre o uso de *big data* no contexto de QoS em Redes Definidas por Software?

A motivação para responder essa questão de pesquisa veio do trabalho de Passos et al. (2018). Na oportunidade, umas das questões de pesquisa busca entender a relação entre objetivos e foco de pesquisa sobre análise de comentários de software. Os objetivos definem a intenção principal do estudo, enquanto o propósito é o principal atributo de interesse.

A intenção dessa pergunta é identificar os objetivos nas pesquisas que abordam o *big data* como recurso auxiliar na QoS em SDN. Ao analisar os estudos, espera-se identificar os propósitos e classificar os objetivos dos estudos primários, e, em seguida, identificar e classificar os focos do trabalho. Assim, será possível realizar uma relação entre objetivo e foco, por exemplo: propor (propósito) um aplicativo para controladores SDN (foco) utilizando *big data* para auxiliar no desempenho da rede.

A classificação das pesquisas quanto aos objetivos foi definida em 3 (três) grupos. **Compreensão**, quando os autores buscam por meio de pesquisa clarear um assunto, ou seja, buscam em fontes primárias sobre o que ele deseja abordar e no artigo tenta explicar, discutir, etc.; **idealização**, quando propõem ou criam uma solução para resolver um problema; **validação**, quando buscam validar soluções existentes, por exemplo, por meio de comparação ou por meio de testes com esse propósito.

Em relação ao foco de pesquisa, foi categorizado em 4 (quatro) grupos. O objetivo é simplificar a compreensão dos estudos e isso não quer dizer que um documento faça parte exclusivamente de um grupo. Partindo dessa definição, o foco foi categorizado da seguinte forma: **características do tráfego**, estão inclusos os estudos que tratavam de assuntos como classificação, caracterização, predição, prioridade e correlação de características; em **segurança da rede** ficaram os que tratavam de detecção e prevenção de ataques ou melhorava, de alguma forma, a segurança da rede; **rota de tráfego** é o grupo que inclui os estudos que tratavam sobre a abordagem para encontrar o melhor caminho para o tráfego; **monitoramento**, por fim, categorizou os estudos que tratavam de alguma forma sobre parâmetros de monitoramento rede.

QP2. Quais os elementos de *big data* que mais aparecem nas pesquisas?

O objetivo dessa pergunta é analisar e identificar quais os elementos de *big data* foram utilizadas na literatura para armazenar, processar e analisar os dados extraídos da rede. A taxonomia de Kune et al. (2016) foi utilizada para nortear a classificação desses estudos.

Segundo Kune et al. (2016), as **dimensões** dos dados são representadas pelos 5Vs do *big data*; ***analytics*** é o processo de analisar os dados usando modelos estatísticos e técnicas de mineração de dados; **tecnologia** é classificada em três partes, a saber, sistema de arquivos, estruturas de computação e ferramentas de análise; **modelos de programação** consistem em dados intensivos, fluxo de computação, processamento em lote, alto desempenho, processamento de consultas e dados orientados por coluna; **segurança** seria o mecanismo para fornecer garantia contra violação enquanto os dados são analisados em vários sistemas distribuídos.

QP3. Quais são os parâmetros de QoS pretendidos em Redes Definidas por Software com a

utilização de *big data*?

A questão quer identificar quais os parâmetros de QoS utilizados quando se faz o uso de *big data* como ferramenta auxiliar em uma rede SDN. Para auxiliar a classificação dos dados extraídos, foi utilizada a taxonomia construída por (SABATA et al., 1997).

Em sua classificação, SABATA et al. (1997), divide QoS em duas categorias principais, métricas e políticas. As métricas especificam parâmetros de QoS quantificáveis e são representadas basicamente por outras três categorias: especificações de desempenho, níveis de segurança e importância relativa. Já as politicas são divididas em nível de serviço e políticas de gestão.

Em resumo, as métricas contemplam os parâmetros relacionados ao desempenho de uma tarefa, como o atraso, volume total, taxa de erro, etc. Além disso, se preocupam com a segurança dos dados e avaliação da importância relativa dos aplicativos que disputam recursos da rede. Por outro lado, as políticas definem ações a serem tomadas pelo gerente da rede em caso de escassez de recursos. O aplicativo pode estar disposto a passar por uma renegociação e aceitar uma qualidade inferior de serviço em vez de ter o serviço negado. As políticas também podem definir comprometimento para uma tarefa ou ainda definir níveis de serviços para o aplicativo "x", disponibilizando garantia ou colocando-o no melhor esforço.

QP4. Quais são as ferramentas de *big data* utilizadas para analisar tráfego em Redes Definidas por Software?

O objetivo dessa questão é identificar e listar ferramentas utilizadas para transmitir, armazenar e processar grandes volumes de dados gerados por uma rede SDN. As respostas dessa pergunta podem ser norteadoras em trabalhos futuros de pesquisadores que desejam implementar algum mecanismo de *big data* em SDN.

QP5. Como é feita a extração dos dados da rede para serem armazenados em *big data*?

A questão quer identificar quais técnicas ou ferramentas são utilizadas para extrair os dados da SDN para serem utilizadas em estruturas de *big data*. Da mesma forma que a Q5, as respostas dessa questão podem orientar, em trabalhos futuros, onde se desejar utilizar *big data* para beneficiar, de alguma forma, as redes definidas por software.

3.1.2 Estratégia de Busca

Neste estudo foram feitas buscas em seis bases de dados digitais: ACM Digital Library, IEEE Xplore, Scopus, Science Direct, SpringerLink e Web of Science. A Scopus engloba trabalhos de outras bases, mas não existe comprovação que seu conteúdo englobe todo referencial sobre a consulta. Assim, as consultas foram realizadas individualmente, visando contemplar uma quantidade maior de trabalhos.

Para delimitar o escopo da busca e indexar os documentos, foram definidas palavras-chave. A escolha dessas palavras é um processo importante, pois é a partir desses termos que

se monta as *strings* de busca, utilizadas nas buscas em bibliotecas digitais. Logo, a criação das *strings* tem impacto direto nos resultados. Os termos utilizados para a busca foram: SDN, *big data* e QoS.

Além das palavras, foram utilizados alguns filtros para refinar a busca como o título, o resumo e as palavras-chave dos trabalhos. Visando estreitar o escopo da busca, foram considerados apenas trabalhos em inglês e na área de pesquisa da Computação.

Para conduzir as buscas nas seis bases, foi definida uma *string* de busca genérica. Em seguida foram feitas adaptações conforme a sintaxe definida por cada uma das bibliotecas digitais. Logo, buscou-se trabalhos científicos em que os termos SDN, *big data* e QoS estivessem contidos nos metadados (Título, Resumo ou Palavras-chave). Na Tabela 3 é possível observar as *strings* definidas para cada uma das bases.

Tabela 3 – Strings de Busca

Base Digital	**String de Busca**	**Qtd.**
String Genérica	("SDN"OR "Software Defined Networking") AND ("BIG DATA") AND ("QoS"OR "Quality of Service")	–
ACM Digital Library	recordAbstract:(("SDN"OR "Software Defined Networking") AND ("BIG DATA") AND ("QoS"OR "Quality of Service")) OR acmdlTitle:(("SDNSoftware Defined Networking") AND ("BIG DATA") AND ("QoSQuality of Service"))	3
IEEE Xplore	("SDN"OR "Software Defined Networking") AND ("BIG DATA") AND ("QoS"OR "Quality of Service")	54
ScienceDirect	TITLE-ABSTR-KEY("SDN"OR "Software Defined Networking") AND ("BIG DATA") AND ("QoS"OR "Quality of Service")	4
Scopus	TITLE-ABS-KEY(("SDN"OR "Software Defined Networking") AND ("BIG DATA") AND ("QoS"OR "Quality of Service"))	65
SpringerLink	("SDN"OR "Software Defined Networking") AND ("BIG DATA") AND ("QoS"OR "Quality of Service")	394
Web of Science	TS=(("SDN"OR "Software Defined Networking") AND ("BIG DATA") AND ("QoS"OR "Quality of Service"))	26
	TOTAL	**547**

Fonte: Autor (2019)

As buscas retornaram 547 trabalhos. Após a conclusão das pesquisas nas bases, iniciou-se a seleção dos trabalhos com base nos critérios e análise dos métodos de seleção.

3.1.3 Critérios de Seleção e Procedimentos de Estudo

A definição de palavras-chave, termos e *strings* não garante um resultado inicial satisfatório, pois podem existir trabalhos que contenham tais termos, mas não se encaixem no contexto pretendido pelo estudo. Para contornar essa situação, foram definidos critérios de inclusão e exclusão, os quais favoreceram uma filtragem eficiente e objetiva, possibilitando a aceitação dos trabalhos considerados relevantes e a rejeição dos demais. Esse filtro foi aplicado ao analisar título, resumo e palavras-chave dos trabalhos retornados nas buscas.

A tecnologia é uma área do conhecimento dinâmica. Os termos SDN e *big data* são recentes, os primeiros relatos são dos anos 2000, por isso, trabalhos anteriores a 2009 foram descartados.

Critérios de Inclusão:

1. Descrevem o uso de Big Data e QoS em SDN
2. No caso de duplicados, o mais recente é selecionado
3. Publicados em workshops, conferências ou revistas
4. Estudos disponíveis na íntegra em formato eletrônico
5. Publicados entre 2009 e 2019

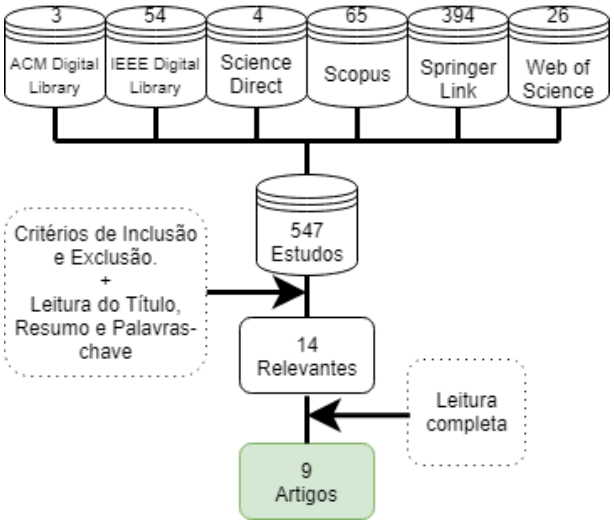
Critérios de Exclusão:

1. Que não descrevem o uso do Big Data e QoS em SDN ou que estejam fora do contexto da pesquisa
2. Duplicados
3. Incompletos
4. Escritos com idioma diferente do Inglês
5. Indisponível

Na Figura 13 é possível observar, de forma resumida, o processo de extração e análise dos estudos.

Ao aplicar os critérios de inclusão e exclusão e realizar a leitura do resumo, palavras-chave e título dos 547 estudos, 14 foram considerados relevantes para leitura completa. Ao final da etapa seguinte, 1 trabalho foi excluído por indisponibilidade da leitura completa e 4 trabalhos foram desclassificados por não descreverem o uso do *big data* e QoS em SDN. Por fim, 9 foram percebidos com forte potencial para responder às perguntas dessa revisão.

Figura 13 – Processo de Busca e Seleção dos Trabalhos



Fonte: Autor (2019)

Os artigos resultantes desse filtro, bem como suas contribuições, serão alvo de debates da Seção 3.2.

3.2 Análise dos Resultados

Nesta Seção, será realizada uma discussão a respeito dos resultados obtidos com a leitura dos 9 artigos considerados potenciais para responder as questões de pesquisa. A tabela 4 mostra os artigos selecionados para esse estudo. Também é possível acessar o link: <https://1drv.ms/x/s!AlpffF3rlub8ioM1AQ8038GmH_lraw?e=ofFunP> para obter detalhes da classificação dos artigos.

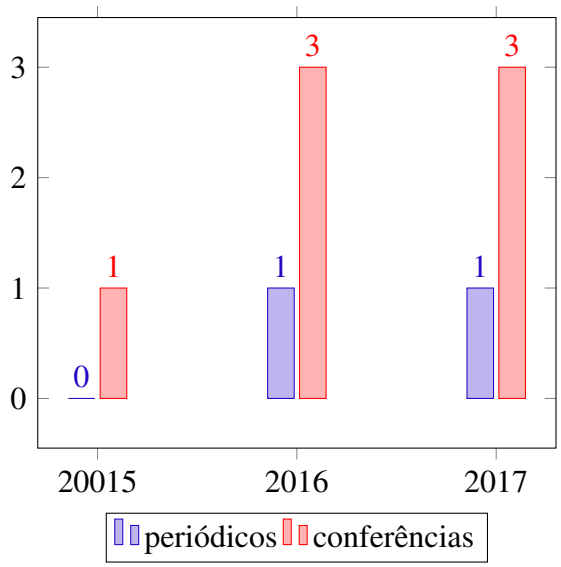
Dos 9 estudos selecionados, (7 - 77,8%) foram publicados em conferências e (2 - 22,2%) em revistas. Do ponto de vista temporal, as publicações foram realizadas a partir de 2015, sendo 2015 o ano com menor número de publicações (1 - 11,1%), seguido de 2016 e 2017, ambos com o mesmo número de estudos (4 - 44,45%).

Tabela 4 – Estudos Selecionados

ID	Título	Referência
A01	Data mining in software defined networking - a survey	(JOSE; NAIR; PAUL, 2017)
A02	Performance evaluation of a Cloud-based QoS support mechanism	(SEMENCIUC et al., 2016)
A03	A tensor-based big data model for QoS improvement in software defined networks	(KUANG et al., 2016)
A04	Uncertainty-driven ensemble forecasting of QoS in Software Defined Networks	(KOLOMVATSOS et al., 2017)
A05	Advanced Control Distributed Processing Architecture (ACDPA) using SDN and Hadoop for identifying the flow characteristics and setting the quality of service(QoS) in the network	(DESAI; S, 2015)
A06	An approach to efficient network design and characterization using SDN and Hadoop	(DESAI; NAGEGOWDA; NINIKRISHNA, 2016)
A07	Applying big data technologies to manage QoS in an SDN	(JAIN et al., 2016)
A08	Programming SDN-Native Big Data Applications: Research Gap Analysis	(ALWASEL et al., 2017)
A09	Secure and QoS aware architecture for cloud using software defined networks and Hadoop	(DESAI; S; T, 2015)

Autor (2019)

Figura 14 – Visão Temporal dos Estudos Selecionados



Fonte: Autor (2019)

Percebe-se, ainda, que os autores Abhijeet Desai, K S Nagegowda e T Ninikrishna, todos naturais da Índia, foram os que mais publicaram sobre o tema; juntos, eles somaram (3 - 33,3%) publicações.

3.2.1 Propósito e Foco de Pesquisa (QP1)

Foram analisados e classificados os objetivos e focos de pesquisa dos trabalhos selecionados. Os resultados consideram que um estudo pode ser categorizado em mais de uma classificação. A saber, um estudo pode explorar o uso de *big data* no contexto de QoS em Redes Definidas por Software para compreender e propor (objetivo) uma arquitetura mais segura (foco).

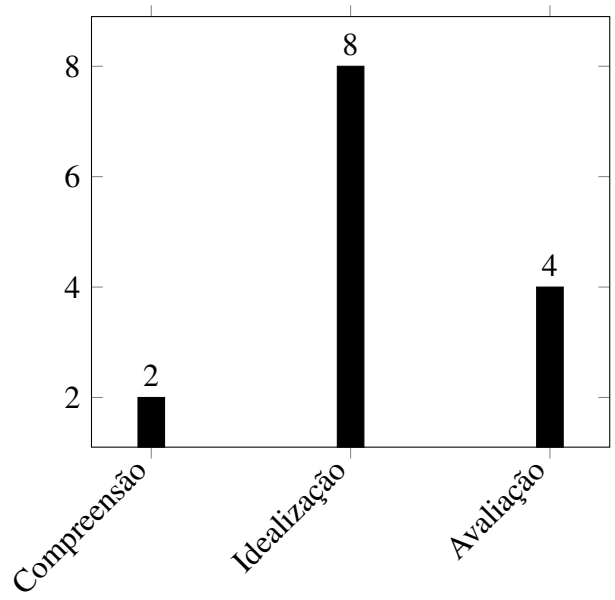
Propósito: A classificação do propósito foi categorizada em três grupos, são eles: compreensão, idealização e avaliação. Dos 9 artigos analisados, (8 - 88,9%) realizaram propostas de soluções envolvendo a tecnologia. Estes se enquadraram na categoria idealização, onde os autores criam ou propõem novas soluções. Apenas (2 - 22,2%) tiveram o propósito de explicar ou discutir sobre o tema e foram enquadrados na categoria de compreensão. Sobre o quesito de avaliação, (4 - 44,4%) realizam testes de validação. A tabela 5 e a figura 15 mostram a distribuição dos artigos nos respectivos grupos.

Tabela 5 – Artigos por Objetivo de Estudo

Grupo	**ID do artigo**
Compreensão	A01, A08
Idealização	A02, A03, A04, A05, A06, A07, A08, A09
Avaliação	A02, A03, A04, A07

Autor (2019)

Figura 15 – Objetivo dos Estudos Selecionados



Fonte: Autor (2019)

Foco: Conforme explicado na Subseção 3.1.1, o foco de pesquisa dos estudos foi categorizado em 4 grupos. A tabela 6 e a figura 16 mostram a distribuição dos artigos nos grupos.

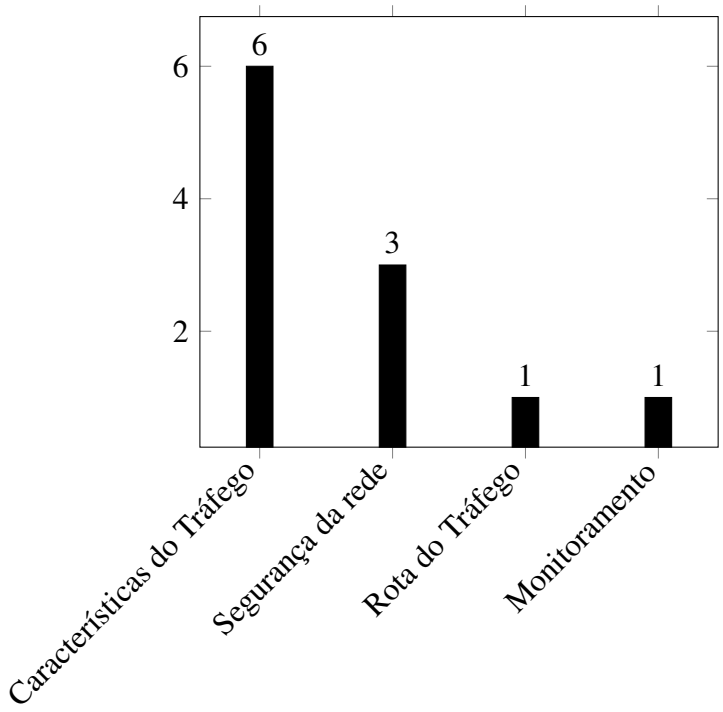
Assim, os resultados se apresentaram da seguinte maneira: características do tráfego é grupo com mais estudos(5 - 55,5%), seguido de segurança(3 - 33,3%). Os demais, tiveram 1 estudo cada (1 - 11,11%).

Tabela 6 – Artigos por Foco de Estudo

Grupo	**ID do artigo**
Características do Tráfego	A01, A02, A05, A07, A08, A09
Segurança da rede	A01, A06, A09
Rota do Tráfego	A03
Monitoramento	A04

Autor (2019)

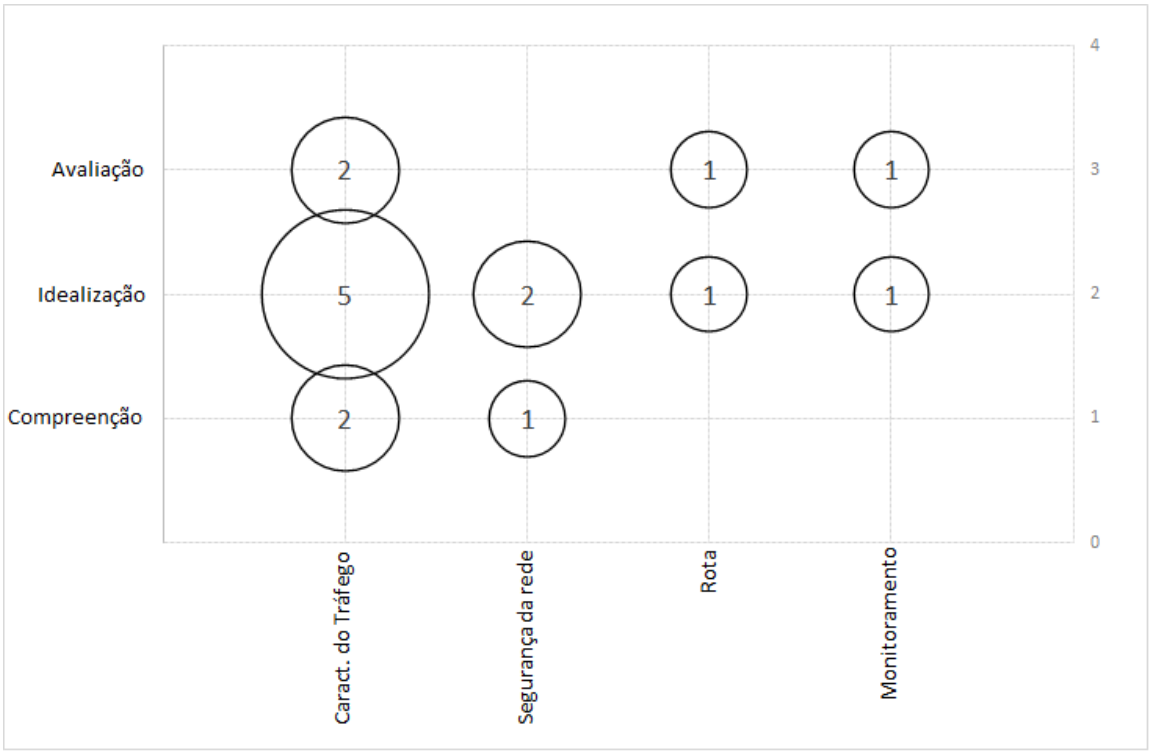
Figura 16 – Foco dos Estudos Selecionados



Fonte: Autor (2019)

Propósito x Foco: A figura 17 é um gráfico de bolhas que mostra a relação entre propósito e foco dos estudos selecionados. Observa-se que a idealização de soluções que abordam características do tráfego e também segurança da rede são as que se mostram mais fortes. Isso é um indício de que os pesquisadores estão propondo soluções em SDN usando *big data* para tratar sobre assuntos como classificação, caracterização, predição, prioridade de tráfego, etc.

Figura 17 – Propósito x Foco



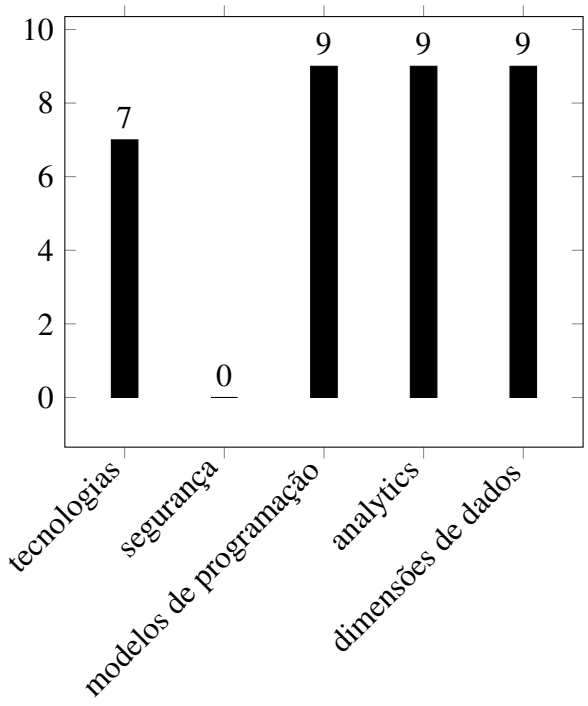
Fonte: Autor (2019)

3.2.2 Elementos de *Big Data* (QP2)

Os elementos de *big data* foram classificados conforme o primeiro nível da taxonomia apresentada em Kune et al. (2016). A taxonomia classifica o *big data* nos seguintes elementos: tecnologia, segurança, modelos de programação, *analytics* e dimensões de dados.

A figura 18 demonstra que todos os trabalhos falam sobre modelos de programação, *analytics* e dimensões de dados. A maioria dos trabalhos (7 - 77,8%) relataram o uso do que Kune et al. (2016) classifica como tecnologia e nenhum trabalho falou sobre a segurança dos dados especificamente. Isso não quer dizer que a segurança dos dados não é um ponto importante, apenas que não foi o foco do estudos selecionados. Em relação à tecnologia, os artigos A03 e A04 não deixam claro o uso desse elemento no documento.

Figura 18 – Elementos de *Big Data*



Fonte: Autor (2019)

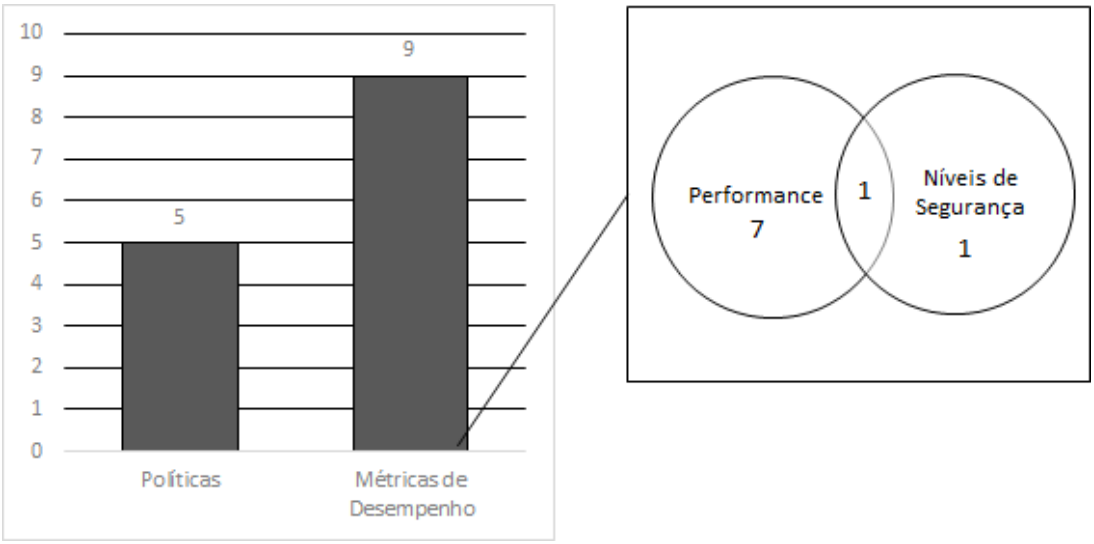
3.2.3 Técnicas de QoS (QP3)

O artigo de SABATA et al. (1997) classifica os parâmetros de QoS em dois grandes grupos: métricas de desempenho e políticas. Analisando os dados de uma maneira geral, percebe-se que todos os artigos possuem foco em métricas de desempenho e apenas (5 - 55,6%) em políticas. É importante destacar que um estudo pode ter sido classificado em ambos os grupos.

As métricas de desempenho são ainda classificadas em: performance, níveis de segurança e importância relativa. Em 2 trabalhos foram tratadas questões relacionadas a nível de segurança e em 7 trabalhos a performance. Nota-se ainda que em 1 trabalho as duas questões foram abordadas.

O grupo políticas está categorizado em: políticas de gerenciamento e nível de serviço. Todos os trabalhos que abordaram técnicas relacionadas à política também foram classificados no subgrupo Políticas de gerenciamento. A figura 19 demonstra de maneira ilustrativa a classificação dos trabalhos na questão de pesquisa 3.

Figura 19 – Técnicas de QoS



Fonte: Autor (2019)

3.2.4 Ferramentas de *Big Data* (QP4)

Durante a leitura dos estudos, um conjunto de ferramentas de *big data* foi mapeado. As ferramentas foram citadas como parte de arquiteturas e/ou soluções propostas ou criadas em integrações com SDN. É importante enfatizar que as ferramentas mapeadas são responsáveis por analisar o tráfego ou fazem parte de soluções com esse objetivo.

As ferramentas Apache Hadoop (HDFS e Mapreduce) e Apache Spark são as ferramentas mais citadas (3 - 33,3 %). Em seguida, estão Algoritmos de Aprendizado de Máquina, Apache Cassandra, Apache Kafka e Aplicativo desenvolvido em Java; todos com (2 - 22,2%) dos relatos. A tabela 7 mostra de uma forma geral as ferramentas.

3.2.5 Extração de Dados (QP5)

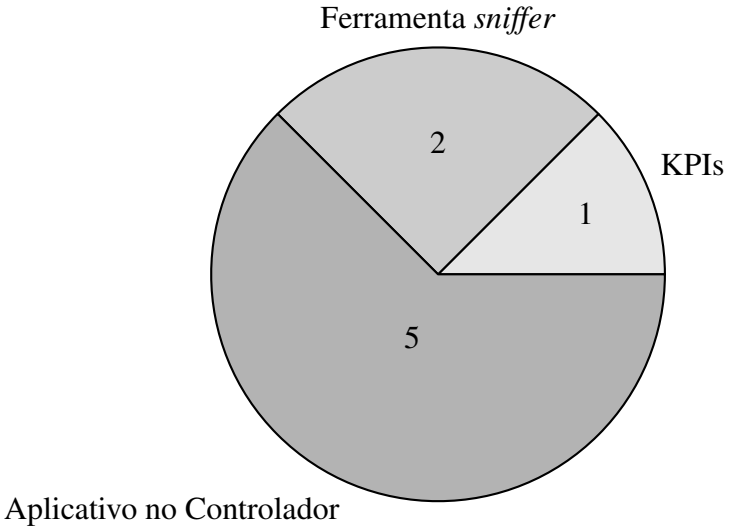
O objetivo dessa questão de pesquisa é identificar como os dados utilizados pelo *big data* para fazer análise são extraídos da rede. O desenvolvimento de aplicativos para controladores de rede foi a maneira mais utilizada, citada em (5 - 55,6 %) trabalhos. Ferramentas *sniffer* foram utilizadas em (2 - 22,2%) documentos para obter essas informações. KPIs (*Key Performance Indicator*) de várias ferramentas e protocolos como SNMP, foi a forma que (1 - 11,1%) trabalho utilizou para capturar dados. Em 1 estudo não foi possível identificar a forma que os dados foram extraídos.

Tabela 7 – Ferramentas de *Big Data*

Ferramentas	**Quantidade**
Apache Hadoop (HDFS e Mapreduce)	3
Apache Spark	3
Machine Learning Algorithms	2
Apache Cassandra	2
Apache Kafka	2
App developed in Java	2
Spearman *algorithm*	1
Apache YARN	1
Apache MESOS	1
Apache Storm	2
R	1
Weka	1
Apache Samza	1

Fonte: Autor (2019)

Figura 20 – Extração de Dados



Fonte: Autor (2019)

3.3 Ameaças à Validade

Os resultados desse trabalho podem ter sido afetados de alguma forma por ameaças à validade, conforme exemplos a seguir:

Strings de busca: embora a pesquisa tenha sido projetada para alcançar o número máximo de obras, é possível que alguns estudos não estejam inclusos por não utilizarem os termos pesquisados. Para mitigar essa ameaça, a *string* de busca genérica passou por um processo de calibração.

Seleção dos estudos primários: em relação à seleção, esta revisão pode não ter mapeado algum trabalho relevante, pela experiência e conhecimento limitada dos pesquisadores. Como forma de mitigar tal viés, três autores participaram da fase de seleção, bem como o trabalho contou com um revisor externo e um revisor interno, além da utilização direta e restrita dos critérios de inclusão e exclusão.

Viés de publicação: embora seja difícil garantir que os principais trabalhos relevantes estejam na pesquisa, as principais bases de dados da computação foram utilizadas para atenuar esse viés.

Questões de pesquisa: as questões de pesquisa podem não cobrir toda área que relaciona *big data*, QoS e SDN. Para lidar com esse risco, pelo menos dois autores analisaram as questões.

Extração de dados: quanto à extração dos dados, este trabalho pode ter sido afetado pelo conhecimento, experiência e perspectiva dos respectivos pesquisadores. Como forma de mitigar tal viés, definiu-se uma questão de pesquisa primária, cinco questões de pesquisa secundárias e critérios claros de inclusão e exclusão. Além disso, a coerência do sistema de classificação elaborada pelos pesquisadores pode significar uma ameaça à validade, pois muitas vezes o conhecimento necessário para elaborá-la só é obtido ao final da seleção (PRETORIUS; BUDGEN, 2008).

Viés de conclusão: Um problema geral, relacionado ao viés de publicação, é a tendência dos pesquisadores em sempre publicar resultados de pesquisa positivos em vez de negativos (KITCHENHAM; CHARTERS, 2007). Entretanto, este risco pode ser considerado mitigado neste estudo, visto que o objetivo desta pesquisa é mapear o que existe na literatura.

3.4 Considerações Finais

Este trabalho fez uma revisão sistemática da literatura sobre o uso de *big data* no aprimoramento de QoS em Redes Definidas por Software. Foram extraídos e analisados dados de 9 trabalhos. Os resultados podem orientar pesquisadores em estudos adicionais que relacionam as áreas de *big data*, QoS e SDN. Para profissionais, técnicas de QoS e *big data* foram catalogadas, além de ferramentas utilizadas em soluções de análise e extração de dados da rede.

Como um possível trabalho futuro, pretende-se combinar evidências identificadas neste trabalho com novas teorias e estudos empíricos a serem desenvolvidos pelos autores, com intuito de criar ou propor novas soluções e ferramentas para apoiar o uso de *big data* no aprimoramento de QoS em SDN.

4

DETCCS - *Decision Engine for Traffic Classification and Control in SDN*

A idealização do DETCCS (*Decision Engine for Traffic Classification and Control in SDN*) surgiu a partir de evidências encontradas na revisão sistemática, descrita no capitulo 3. A revisão buscou entender como o *big data* tem sido explorado para auxiliar o QoS em Redes Definidas por Software. Desta forma, essa pesquisa mostrou indícios que há interesse por parte da comunidade científica na idealização de soluções para melhorar, dentre outros pontos, a classificação de tráfego, possibilitou listar ferramentas de *big data* comumente utilizadas e conhecer a forma como os dados são extraídos.

Baseando-se nos resultados obtidos na revisão, bem como nas experiências relatadas nos trabalhos selecionados, buscou-se projetar uma solução que permita a coleta de informações referentes aos fluxos e suas estatísticas nos dispositivos de comutação através do controlador de rede utilizando o protocolo *OpenFlow*. A partir daí, esse grande volume de dados será analisado por uma estrutura de *big data* para compreensão de padrões de tráfego e assim seja possível definir classes que possibilitem a aplicação de políticas que venham a contribuir na qualidade do serviço e potencialize o aproveitamento dos recursos de redes multiserviços.

Os resultados da revisão sistemática despertaram o interesse e remeteram à possibilidade de montar um cenário integrando ferramentas que juntas possibilitassem alcançar o objetivo desejado. A junção dessas ferramentas trouxe a solução mais próxima do termo "mecanismo". Segundo o dicionário online Dicio (2020), mecanismo é a junção de peças dispostas de modo a fazer com que algo funcione e isso é exatamente o que esse trabalho se propõe.

Os principais diferencias do DETCCS estão na abrangência da solução, que vai desde a coleta das estatísticas de fluxos até a aplicação de regras de QoS e na flexibilização de possibilidades que permitem avanços importantes em pesquisas futuras. Buscou-se incluir ferramentas que trabalhem com arquitetura distribuída visando a escalabilidade; ferramentas de código aberto para aumentar as possibilidades de customização e redução de custo; ferramentas que possibilitem analisar o cenário da rede através de dados existentes ou em tempo real através

de *streaming*; que permitem a utilização de aprendizado de máquina sem desprender muito esforço; que permitem flexibilizar os parâmetros das análises; que permitem o desenvolvimento em diversas plataformas, bem como o jeito como os dados serão apresentados ou como o administrador fará a interação com a rede; que possibilitem a coleta de estatísticas de fluxos diretamente no controlador e utilizem recursos nativos do OpenFlow para aplicar políticas de QoS.

A Figura 21 ilustra a organização dos componentes do DETCCS. Na camada de controle é possível observar a presença de dois módulos, um para coleta de estatísticas (*"StatisticsCollector"*) e outro para realizar gestão automatizada de QoS (*"QoSManager"*). Em paralelo à camada de aplicação, há uma estrutura de *big data* que dará apoio ao *QoSManager* e a outras possíveis aplicações que possam beneficiar-se dos *insights* obtidos. Ainda é possível observar uma API *(pplication Programming Interface)* que possibilita o administrador de rede agendar e obter status de tarefas criadas a partir de análises desenvolvidas no módulo de análise *"SparkModule"*.

A comunicação entre a camada de aplicação e a camada de controle de uma rede SDN é um tema bastante discutido e ainda sem definição na literatura. Não existe uma padronização para a interface *northbound* (NUNES et al., 2014; MACEDO et al., 2015). Portanto, a interoperabilidade entre o *"StatisticsCollector"* e a estrutura de *big data* foi um desafio discutido neste trabalho, cuja superação foi essencial para a efetividade da abordagem aqui apresentada.

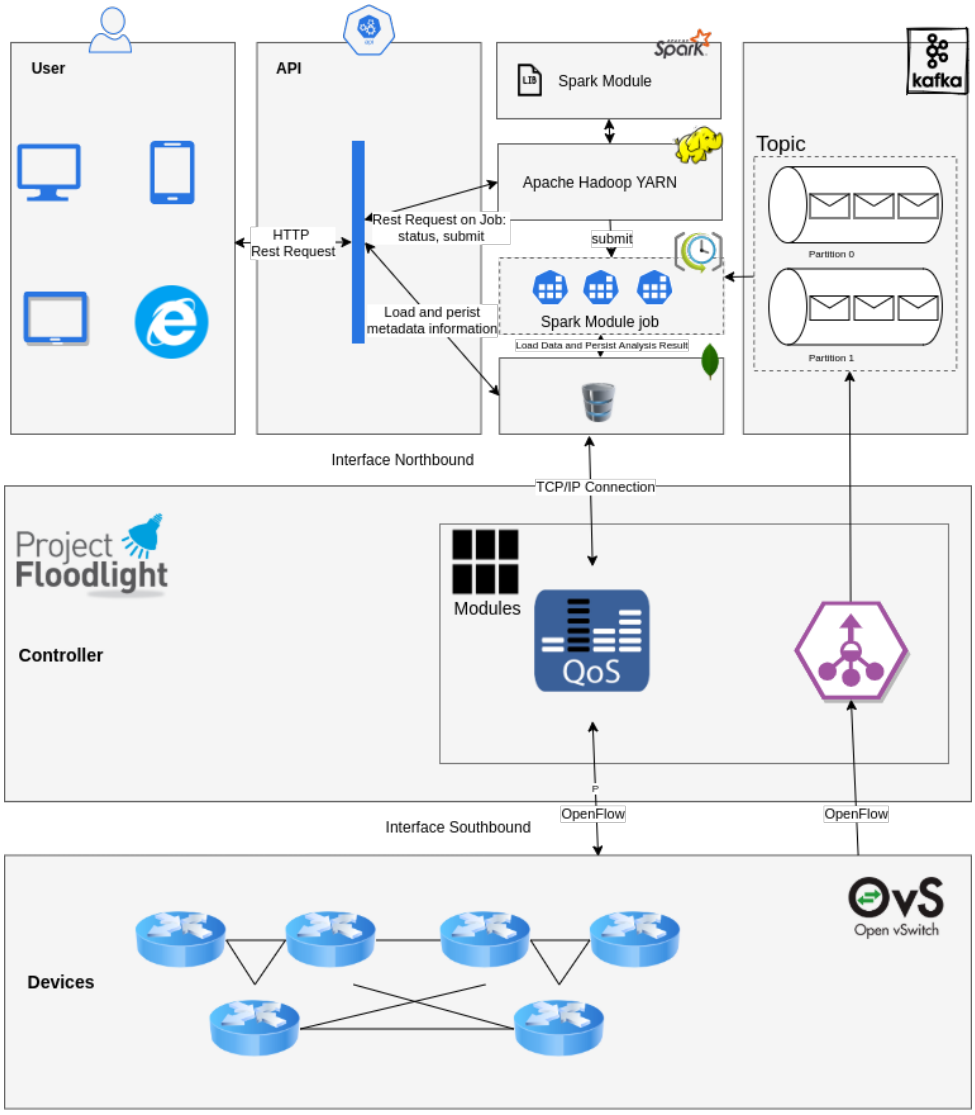
As próximas sessões demonstrarão detalhes dos componentes, ferramentas e como eles estão interligados.

4.1 Interface *Southbound* e Controlador de Rede

As interfaces *southbound* são padronizadas e cuidam da interação entre o controlador, os elementos de rede e as interfaces programáveis nos elementos de borda. O protocolo OpenFlow é um dos exemplos mais conhecidos de uma API *southbound* no contexto de SDN, possui uma vasta comunidade, uma boa documentação e é bem aceito no mercado. Esses motivos impulsionaram a escolha para integrar o DETCCS. A versão do protocolo utilizada no projeto é a versão 1.3.

Atualmente, existem vários controladores de rede que suportam o protocolo OpenFlow. Em sua dissertação, Ferreira et al. (2016) faz um comparativo entre os principais controladores, mais especificamente o POX, NOX, ONOS, Beacon, Maestro, Ryu, Trema, FlowER, FloodLight e o OpenDaylight. O controlador escolhido para esse trabalho foi o FloodLight. Está entre as características do FloodLight, ser *open source*(código aberto), baseado na linguagem de programação Java, com licença Apache 2.0 e suportado por uma comunidade extensa de desenvolvedores (FLOODLIGHT, 2019). Além dessas características, um fator importante para a escolha desse controlador foi o alinhamento colaborativo entre pesquisadores do ELAN(*Experimental Laboratory on Computer Networks*) da Universidade Federal de Sergipe em trabalhos que estão

Figura 21 – Arquitetura da Solução



Fonte: Autor (2020)

sendo desenvolvidos e também a expertise acumulada em trabalhos já publicados, tais como (CARVALHO, 2017; SILVA et al., 2016).

4.2 Coletor de Estatísticas

Para compreender o módulo de coleta das estatísticas dos fluxos implementada no DETCCS, é importante uma pausa para relembrar e entender o *pipeline* OpenFlow em cada comutador. Os comutadores possuem pelo menos uma tabela de fluxo e cada tabela de fluxo contém várias entradas de fluxo. A partir daí, quando um pacote chega ao *switch* ele é comparado com as entradas de fluxos da tabela. Se uma entrada correspondente for encontrada, o conjunto de instruções incluído nessa entrada de fluxo será executada, caso contrário, o pacote pode ser enviado para o controlador pelo OpenFlow *channel* através de uma mensagem assíncrona de

entrada de pacote (*Packet-In*), ou pode ser descartado ou ainda continuar com a próxima tabela de fluxo. Além disso, é possível que o controlador seja informado através de uma mensagem assíncrona (*Flow-Removed*) sobre a remoção de uma entrada de fluxo com o conjunto de sinalizadores OFPFF_SEND_FLOW_REM de uma tabela (FOUNDATION, 2012).

Partindo desse entendimento, um módulo denominado *"StatisticsCollector"* foi desenvolvido no controlador FloodLight para coletar estatísticas de fluxos dos comutadores. A coleta de estatísticas realizada pelo módulo segue a especificação de mensagens assíncronas *Flow-Removed* do OpenFlow. Desta forma, toda vez que houver solicitação de exclusão de fluxo partindo do controlador ou quando o tempo limite do fluxo for excedido no comutador, uma mensagem *Flow-Removed* será enviada ao controlador com todas as informações do fluxo. Cada mensagem de fluxo removida contém uma descrição completa do fluxo de entrada, o motivo da remoção (expiração ou exclusão), a duração até o momento da remoção e as estatísticas até o momento da remoção.

O mecanismo de expiração do fluxo é executado pelo comutador independentemente do controlador e é realizado com base no estado e na configuração das entradas de fluxo. Cada entrada de fluxo possui dois campos de configuração (*idle_timeout* e o *hard_timeout*) associado. Se qualquer valor for diferente de zero, o comutador deve observar o tempo a partir da chegada do fluxo. Um campo *hard_timeout* diferente de zero faz com que a entrada do fluxo seja removida após o número especificado de segundos, independentemente de quantos pacotes corresponderem. Já o campo *idle_timeout* faz com que a entrada do fluxo seja removida quando passar o número de segundos predefinidos sem a chegada de nenhum pacote referente ao fluxo (FOUNDATION, 2012).

Para que a assertividade quanto às características reais do fluxo seja maior, o controlador FloodLight foi configurado para que o parâmetro *idle_timeout* fosse definido por padrão em 5 segundos em todas as entradas. Isso irá permitir que após 5 segundos de inatividade o fluxo seja removido da tabela e o controlador será notificado com todas as informações. O Código 1 é um exemplo de informações coletada através de uma massagem *Flow-Removed* pelo módulo *"StatisticsCollector"*. Ao receber a mensagem do comutador, o módulo *"StatisticsCollector"* transforma a mensagem em formato JSON (*JavaScript Object Notation*). Esse formato possui sintaxe simples, de fácil aprendizado e implementação, além de ser leve e fácil de ler. O formato JSON foi escolhido para facilitar a integração com outras camadas ou possíveis aplicativos que desejem fazer uso desses dados.

Observa-se que a partir da linha 2 do código 1, no primeiro nível do JSON, informações importantes sobre a mensagem *openflow* são obtidas. Nesse primeiro nível, informações como a instancia do *openflow* no *switch*, o *Datapath Identifier* (dpid), o identificador da tabela (*tbid*), tipo da mensagem (*type*), prioridade da mensagem (*priority*), campos de configuração para remoção do fluxo na tabela de fluxo (*idle_time_out* e *hard_time_out*), a hora em que o fluxo termina(*msg_arrival_time*). É interessante destacar que o valor do campo *msg_arrival_time* é

Código 1 – JSON com Informações do Fluxo

```
{
    "dpid" : "00:00:00:00:00:00:00:01",
    "tbid" : 0,
    "type" : "FLOW_REMOVED",
    "priority" : 1,
    "idle_time_out" : 0,
    "hard_time_out" : 0,
    "msg_arrival_time" :"2017-04-27T14:28:41.000+0000",
    "match" : {
        "eth_src" : "74:c7:af:8d:b2:eb",
        "eth_dst" : "94:8a:51:eb:73:59",
        "ip_src" : "10.200.7.195",
        "ip_dst" : "169.46.131.128",
        "port_src" : 48055,
        "port_dst" : 443,
        "protocol_number" : 6,
        "eth_type" : 2048,
        "ip_dscp" : 0
    },
    "counter" : {
        "packet_in_count" : 9,
        "byte_in_count" : 1048,
        "duration_sec" : "19.743482",
        "duration_n_sec" : "19743482000",
        "duration_sec_total" : "19.743482",
        "duration_n_total" : "19743482000",
        "length_avg_packet" : "116.44444444444444440000",
        "bytes_per_sec" : "53.080809150077985200000",
        "packet_per_sec" : "0.45584664348000000000"
    }
}
```

Fonte: Autor (2020)

uma informação obtida pelo módulo desenvolvido quando o fluxo chega ao controlador e não no momento exato em que ele é removido da tabela de fluxo do *switch*, ou seja, é uma informação aproximada.

No segundo nível do código 1 é possível observar informações sobre os campos de correspondências da mensagem *openflow*, os campos *metch*. São exemplos de campos de correspondência: o IP de origem e destino (*ip_src* e *ip_dst*), portas de origem e destino (*port_src* e *port_dst*), endereço MAC *ethernet* de origem e destino (*eth_src* e *eth_dst*), número do protocolo (*protocol_number*), tipo do quadro *ethernet* (*eth_type*), O *DiffServ Code Point (ip_dscp)*.

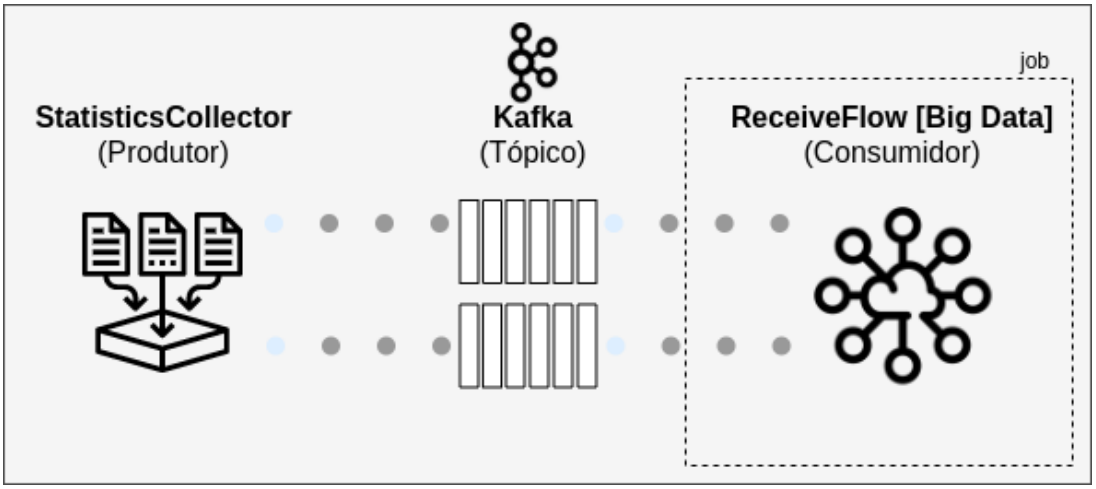
Em relação às estatísticas, pode-se observar a partir da linha 20 do código 1 os campos pacotes recebidos (*packet_in_count*), bytes recebidos (*byte_in_count*), duração em segundos (*duration_sec*), duração em nanossegundos (*duration_n_sec*) e outras derivações desses campos. Os contadores são atualizados por fluxo, conforme explicado na subseção 2.4

Essas informações são muito importantes para análise do comportamento da rede. O módulo *"StatisticsCollector"* envia os dados coletados em tempo real à estrutura de *big data* por meio de *streaming*. Essa etapa será detalhada na próxima sessão.

4.3 Interoperabilidade na Coleta de Estatísticas

Para lidar com interoperabilidade entre o módulo *"StatisticsCollector"* e a estrutura de *big data*, utilizou-se uma plataforma de *streaming* distribuída, denominada Apache Kafka. O Kafka é uma plataforma *open-source* de processamento de *streams* desenvolvida pela *Apache Software Foundation*. De forma resumida, o fluxo básico da ferramenta é dividido em três momentos: (1) criação da mensagem, (2) registro em um tópico, (3) consumo da mensagem. A ilustração 22 demonstra de forma simplificada o fluxo do DETCCS utilizando o Apache Kafka.

Figura 22 – Fluxo Básico do Kafka



Fonte: Autor (2020)

Segundo KAFKA (2020), a ferramenta Apache Kafka é recomendada nos casos em que a construção de pipelines de dados de *streaming* em tempo real entre sistemas ou aplicativos de maneira confiável é necessária ou para construção de aplicativos de *streaming* em tempo real que transformam ou reagem aos fluxos de dados.

Aplicando o entendimento básico do Kafka no DETCCS, o módulo *"StatisticsCollector"* funciona como o produtor de mensagem a um determinado tópico. Um exemplo de mensagem é o JSON já observado no código 1. O receptor poderia ser qualquer aplicação assinante desse tópico. Na solução, esse receptor é um *job* presente na estrutura de *big data* responsável por recepcionar e armazenar a mensagem.

O Kafka pode ser executado em uma única instância em uma máquina, quando isso acontece, dá-se o nome de *broker* kafka. Ainda é possível ser executado em várias instâncias e em várias máquinas, nesse caso o ambiente é chamado de *cluster* Kafka. A comunicação entre os clientes e os servidores é feita com um protocolo TCP, independente da linguagem utilizada (KAFKA, 2020).

A escolha do Kafka para compor a estrutura do DETCCS deu-se por ser uma ferramenta escalável, resiliente e atende as necessidades de interoperabilidade impostas pela solução na transferência de um grande volume de dados em tempo real entre diferentes sistemas.

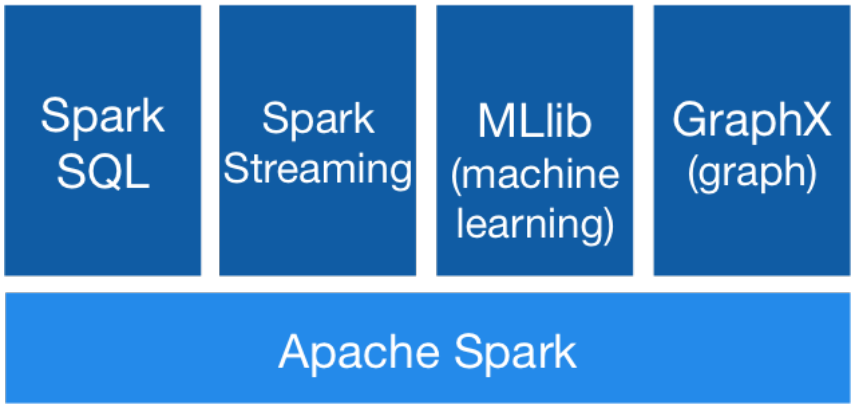
4.4 *Big Data* e Análise de Dados

Analisar o grande volume de dados produzidos por uma SDN e obter *insights* em um espaço curto de tempo é um dos objetivos desse trabalho. Portanto, a utilização de uma estrutura de *big data* é justificada por lidar com dados massivos, gerados em uma velocidade crescente e que precisam ser processadas e analisadas com segurança e confiabilidade de modo a extrair conhecimento.

No Capitulo 3, uma das questões discutidas foram as ferramentas de *big data*. A questão de pesquisa (QP4) da revisão sistemática indagou quais foram as ferramentas de *big data* utilizadas para analisar tráfego em Redes Definidas por Software nos trabalhos selecionados. As ferramentas Apache Hadoop e Apache Spark foram as mais utilizadas. No entanto, para esse trabalho o Apache Spark foi escolhido por possuir uma pilha de bibliotecas para *streaming*, aprendizado de máquina e outras funcionalidades que poderiam agregar no resultado pretendido. Os principais componentes do Spark podem ser vistos na figura 23.

O Apache Spark é definido como um mecanismo de análise unificada extremamente rápido para *big data*, aprendizado de máquina e processamento de dados em larga escala (DATABRICKS, 2020; SPARK, 2020). O Spark pode ser executado em *clusters* do Hadoop por meio do modo independente YARN e também pode processar dados no HDFS(*Hadoop Distributed File System*), HBase, Cassandra, Hive e em qualquer Hadoop *InputFormat*, conforme

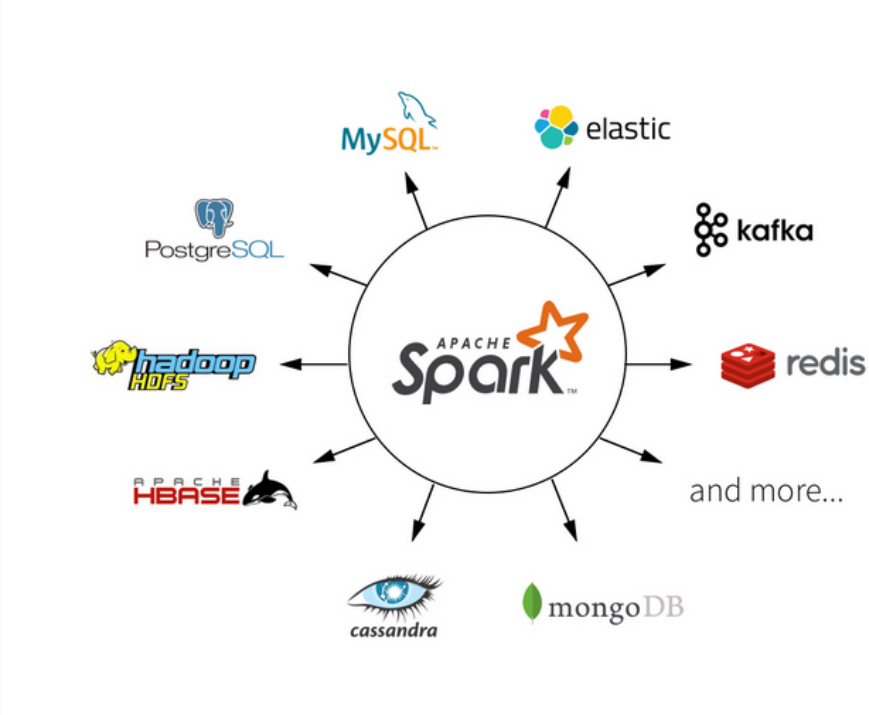
Figura 23 – Componentes do Apache Spark



Fonte: (SPARK, 2020)

figura 24. O Spark chega a executar 100 vezes mais rápido que o MapReduce, quando se trata de cargas de trabalho em memória, e 10 vezes mais rápido quando se trata de cargas de trabalho em disco (SPARK, 2020).

Figura 24 – Fontes de Dados do Apache Spark



Fonte: (DATABRICKS, 2020)

Outra grande vantagem do Spark é que todos os componentes funcionam integrados na própria ferramenta, como o Spark *Streamming*, o Spark SQL e o GraphX, como mostra a figura 23, diferentemente do Hadoop, onde é necessário utilizar ferramentas que se integram a ele, mas que são distribuídas separadamente, como o Apache Hive. Além disso, outro aspecto importante

é que ele permite a programação em três linguagens: Java, Scala e Python. Na solução proposta, utilizou-se a linguagem de programação Java (PENCHIKALA, 2020).

Quando o módulo *"StatisticsCollector"* envia os dados por meio do Kafka para a estrutura de *big data*, utiliza-se o Spark *Streamming* para analisar em tempo real ou armazenar no banco de dados MongoDB. O MongoDB é um banco de dados distribuído, baseado em documentos e de propósito geral, fornece alto desempenho, alta disponibilidade e fácil escalabilidade. O MongoDB foi escolhido pelas características já citadas e pela simplicidade de uso dentro da estrutura do Spark.

4.4.1 Análise de Dados com *SparkModule*

O *SparkModule* é um módulo desenvolvido na linguagem Java utilizando as bibliotecas do Apache Spark para formulação de regras de negócio. Em resumo, o *SparkModule* contém todas as análises que serão transformadas em tarefas agendadas pelo administrador da rede. Por exemplo, o administrador gostaria de analisar os fluxos da rede coletados nos últimos 30 dias utilizando algoritmos de aprendizagem de máquina não supervisionado e repetir essa análise a cada 24 horas. Nesse caso, a análise é desenvolvida e compilada no "*SparkModule*"e agendada através da API para ser executada a cada 24 horas, passando os parâmetros desejados.

Uma análise possui parâmetros importantes que podem indicar a origem dos dados a serem analisados, onde os resultados serão armazenados, o período de análise e os dados de quais comutadores serão utilizados. Além desses, existem parâmetros específicos de cada análise que serão discutidos no experimento.

A sessão 4.5 irá mostrar com um nível maior de detalhes um exemplo prático de um agendamento de tarefa na qual uma análise do *SparkModule* e seus parâmetros é utilizada.

4.5 API e Gestão de Agendamentos de Tarefas

Para manter o máximo de aproveitamento dos recursos da rede é necessário que o gestor acompanhe as mudanças no comportamento do tráfego e aplique políticas que se enquadrem melhor em cada momento. Para alcançar esse objetivo, é preciso ter uma visão do comportamento da rede em relação ao tempo. Portanto, analisar os dados de tempos em tempos torna o cenário mais claro ao administrador e permite que decisões melhores sejam adotadas.

Com o propósito de sanar essa demanda, foi desenvolvido um agendador de tarefas e uma API que garante ao administrador a flexibilidade de agendar análises em períodos desejados e com uma boa elasticidade nos parâmetros de cada análise. Isso assegura que os dados de tráfego da rede sejam analisados por várias vertentes e que o administrador não fique preso ao código fonte da análise desenvolvida no *SparkModule*. Com a API desenvolvida, garante-se uma flexibilidade de realizar essas tarefas através de aplicações que podem ser desenvolvidas em

diversas plataformas, como por exemplo, mobile, web ou desktops, etc.

O código 2 mostra com detalhes um agendamento para uma análise através da API. É possível observar alguns parâmetros importantes no agendamento, como por exemplo a identificação da análise (*analysisId*), previamente cadastrada, frequência com que a análise será executada (*frequency* em milissegundos) e diversos parâmetros específicos em (*analysisArgs*) criados através de (*key–value*) para análise em questão. Por exemplo, o parâmetro *switches* indica quais pontos da rede serão analisados, enquanto os parâmetros *database* e *flow* indicam, respectivamente, qual banco de dados e a coleção que será analisada.

Além de realizar o cadastro e a alteração de agendamento de tarefas, a API permite listar todas ou apenas uma tarefa já agendadas através de seu identificador, deletar e alterar o status para ativa ou inativa. Essas operações simples de consulta, atualização e destruição também podem ser realizadas nas análises.

Código 2 – Agendamento de Análise

```
{
  "schedulerId": "adasdnSchedule_1594845187447",
  "analysisId": "AnalyseFlowByDuration",
  "description": "Agendamento de Teste 01",
  "analysisArgs": [
    {
      "name": "k",
      "value": "4"
    },
    {
      "name": "database",
      "value": "adasdn"
    },
    {
      "name": "collection",
      "value": "flow"
    },
    {
      "name": "matchs",
      "value": "match.ip_src,match.ip_dst"
    },
    {
      "name": "individualAnalysis",
      "value": "true"
    },
    {
      "name": "switches",
      "value":
      ↪  "00:00:00:00:00:00:00:01,00:00:00:00:00:00:00:02,00:00:00:00:00:00:00:03"
    },
    {
      "name": "period",
      "value": "2592000000"
    }
  ],
  "frequency": 3600000,
  "lastExecution": 1600886031027,
  "enabled": true
}
```

Fonte: Autor (2020)

4.6 Gestão de QoS

Os resultados das análises desenvolvidas no *SparkModule* servirão de apoio para criação de políticas de qualidade de serviço nos comutadores da SDN. As politicas serão criadas através da API disponibilizada ao administrador e enviada aos comutadores através do protocolo OpenFlow. Após as politicas serem aplicadas aos comutadores, cada novo fluxo será analisado pelo módulo *QoSManager* e ele decidirá qual a política mais apropriada para o novo fluxo.

As políticas podem ser aplicadas a um ou mais comutadores da rede e associadas a uma classe de tráfego proveniente dos resultados das análises. Por exemplo, se o comportamento do tráfego for analisado por algoritmos de aprendizado de máquina não supervisionado de agrupamento, o resultado será o surgimento de grupos formados por fluxos de comportamentos semelhantes, nesse caso, o administrador poderá observar o comportamento de cada um, criar regras de QoS e associá-las a esses grupos.

A tabela 8 demonstra um exemplo hipotético em que três grupos foram criados baseados nos resultados das análises do *SparkModule* e regras de QoS foram associadas a eles. No exemplo, é possível observar regras criadas a partir de recursos nativos do OpenFlow, no caso, a limitação de taxas de transferência com medidores. Medidores do OpenFlow podem ser vistos com mais detalhes na subseção 2.1.8. Seguindo o exemplo da tabela 8, após criadas as regras de QoS, os fluxos que se enquadrarem no Grupo 1 terão uma taxa de transferência limitada a 5 MB/s. Diferente dos fluxos que condizem com o grupo 2, esses terão uma taxa de transferência reduzida a 2 MB/s e assim funciona para os demais grupos. A partir daí, os campos de correspondência de cada novo fluxo serão analisados e, no caso do fluxo pertencer a algum grupo, seguirá as regras de QoS associadas a ele.

Tabela 8 – Exemplo de Regras Básicas de QoS

Resutado da Análise	**Limitação de taxa**
Grupo 1	5 MB/s
Grupo 2	3 MB/s
Grupo 3	2 MB/s

Autor (2020)

5

Estudos de Caso

Este capítulo apresenta os dois estudos de caso realizados nessa dissertação. O primeiro consiste em um estudo de caso básico em formato de prova de conceito, realizado no início dessa pesquisa, cujos resultados possibilitaram o prosseguimento do trabalho. O segundo é um estudo de caso que usa dados coletados de uma rede real e tem o objetivo de extrair valor dos dados analisando maior número de caraterísticas de fluxos para extrair valor e auxiliar na criação de regras de QoS em uma SDN emulada. Segundo Yin (2015), um estudo de caso é um tipo de experimento planejado que objetiva analisar fenômenos de um determinado ambiente, por meio de múltiplas fontes de evidências.

5.1 Planejamento

O planejamento é uma ferramenta importante para melhoria de processo onde o resultado depende de diversas variáveis ou da combinação destas. O sucesso de um planejamento de experimentos dependerá, em grande parte, da forma com que este é estruturado e como será realizado. Entender claramente quais são os objetivos de realizar um experimento é necessário antes de qualquer ação para executá-lo (VIEIRA, 2007).

Faz parte do objetivo geral dessa dissertação analisar e compreender os padrões de tráfego de redes SDN. Em outras palavras, em uma das etapas do trabalho pretende-se realizar descoberta de conhecimento ao analisar dados extraídos da rede. Desta forma, o planejamento dos estudos de casos foram **baseados** no processo KDD(*Knowledge Discovery in Databases*), que significa "descoberta de conhecimento em bancos de dados"(AZEVEDO; SANTOS, 2008). O objetivo principal do KDD converge com um dos objetivos desse trabalho que é a extrair conhecimento de grandes bases de dados. Para descoberta de conhecimento através do KDD, é necessário seguir cinco etapas que serão detalhadas a seguir e ilustradas na figura 25.

1. **Seleção** - Este estágio consiste em criar um conjunto de dados ou focar em um subconjunto

Figura 25 – Etapas do Processo KDD

Fonte: (SANDRI, 2008)

de variáveis ou amostras de dados, nas quais a descoberta deve ser realizada.

2. **Pré-processamento** - Esta etapa consiste na limpeza e pré-processamento dos dados para obter dados consistentes.

3. **Transformação** - Esta etapa consiste na transformação dos dados usando métodos de redução ou transformação de dimensionalidade.

4. ***Data Mining*** - Esta etapa consiste na busca de padrões de interesse em uma determinada forma representacional, dependendo do objetivo de mineração de dados.

5. **Interpretação/Avaliação** - Esta etapa consiste na interpretação e avaliação da descoberta de padrões.

Além dos passos descritos para geração de conhecimento, foi utilizado o software Iperf para gerar e medir a limitação da taxa de transferência criadas a partir de medidores no OpenFlow com o módulo desenvolvido *QoSManager*.

5.1.1 Instrumentação

Para realizar os estudos de caso, utilizou-se a estrutura do Laboratório Experimental de Redes de Computadores (ELAN), da Universidade Federal de Sergipe. Dois servidores foram disponibilizados para instalação dos recursos de software listados na tabela 9. O primeiro servidor, aqui denominado de SRV1, possui 8GB de memória RAM e processador Intel(R) Core(TM) i5-4440 de 3.10GHz de quatro núcleos. O segundo servidor, aqui denominado de SRV2, possui 16GB de memória RAM e processador Intel(R) Xeon(R) CPU E5430 de 2.66GHz de quatro núcleos. A estrutura de *big data* foi instalada no SRV2 e o controlador de rede no SRV1.

Embora a solução possa ser empregada em ambientes reais, nesta dissertação foi utilizado um ambiente SDN emulado para caracterização dos elementos que compõem a topologia de rede. Desta forma, utilizou-se o Mininet como emulador dos componentes da SDN. No entanto, outros recursos utilizados na solução foram instalados em máquinas reais. A tabela 9 mostra de forma detalhada os recursos utilizados.

Tabela 9 – Recursos de Software

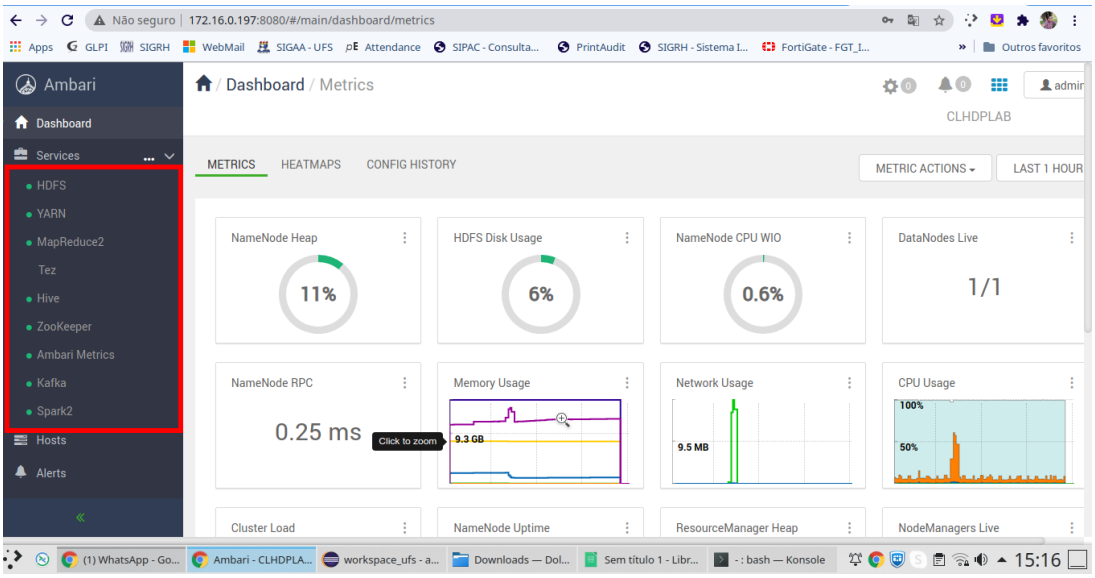
DESCRIÇÃO	**VERSÃO**	**TIPO**
Mininet	2.2.2	Emulador de rede
Open vSwitch	2.13.1	Comutador de rede
Kafka	2.13	Plataforma de *streamming*
Floodlight	1.2	Controlador SDN
MongoDB	4.4.0	DB NoSQL orientado a documentos
Apache Spark	2.11	Mecanismo de análise unificado para processamento de dados em grande escala
Iperf	2.0.13	Ferramenta de geração e medição de Tráfego TCP e UDP
D-ITG	2.8.1	Ferramenta de geração e medição de Tráfego

Fonte: Autor (2020)

5.1.2 Operação

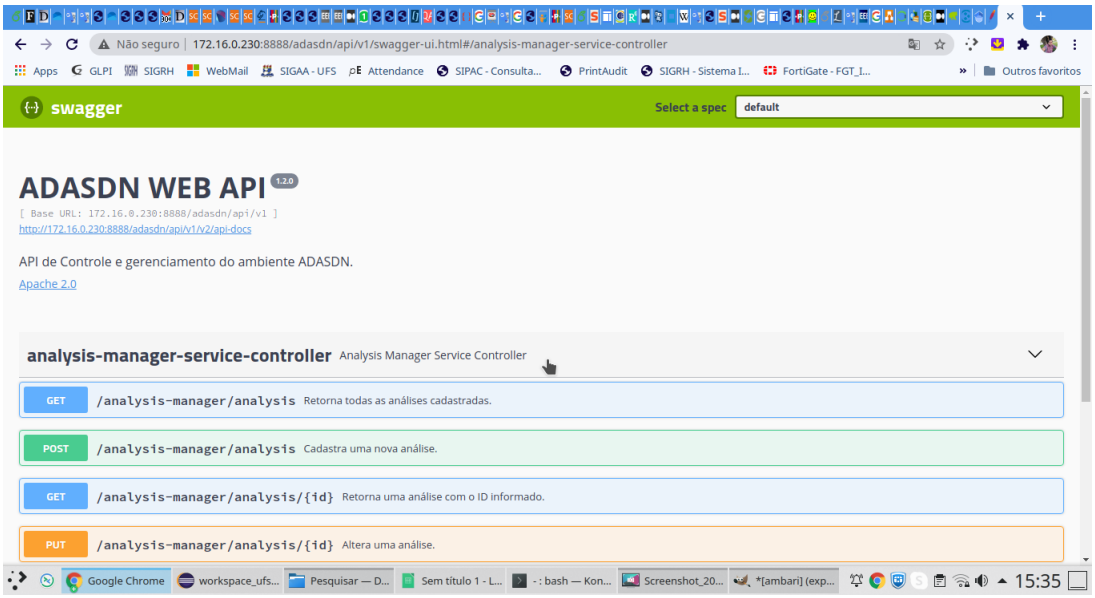
Para a execução dos estudos de caso, foi necessário a preparação do mecanismo no ambiente disponibilizado pela UFS, o ELAN. No SRV1, foi instalado o controlador de rede Floodlight, a API e o banco de dados MongoDB. No SRV2, foi instalado o Apache Spark, o Apache Yarn e o Kafka, todos gerenciados pelo Ambari (2020). O ambiente gerenciado pelo Ambari já era utilizado em outras pesquisas do laboratório, apenas adicionamos mais ferramentas. O projeto Apache Ambari visa tornar o gerenciamento e monitoramento de *clusters* do Hadoop mais simples (AMBARI, 2020). As imagens 26, 27 e 28 mostram o ambiente Ambari configurado, o print da API desenvolvida e o print da tela inicial do controlador Floodlight, respectivamente.

Figura 26 – Captura de Tela do Ambiente Ambari



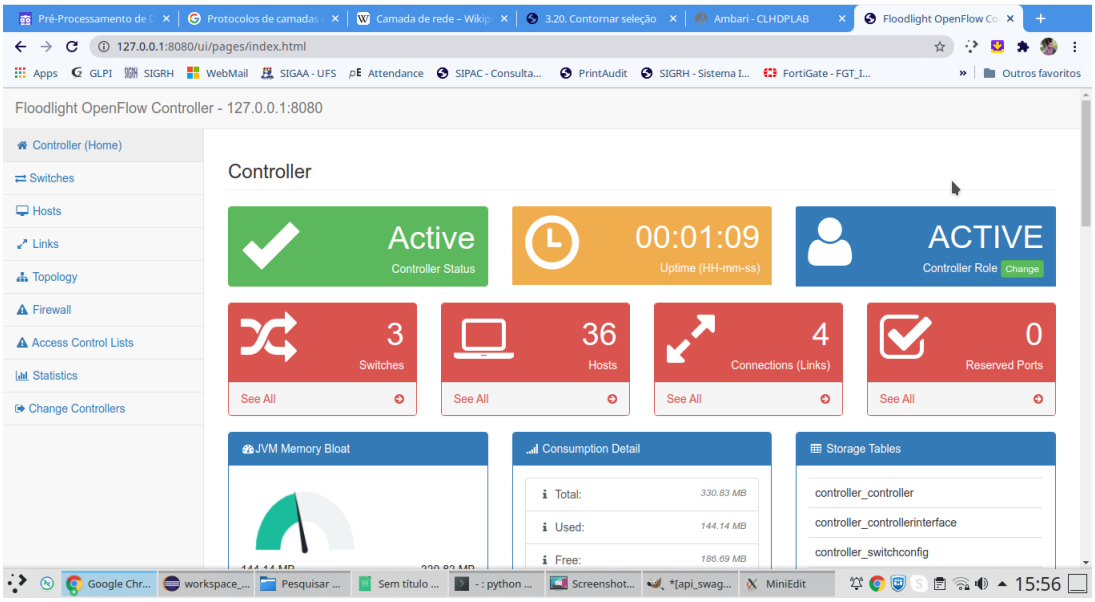
Fonte: Autor (2020)

Figura 27 – Captura de Tela do Tela API



Fonte: Autor (2020)

Figura 28 – Captura de Tela do Tela Controlador Floodlight



Fonte: Autor (2020)

5.2 Execução da Prova de Conceito

Nesta Seção, será realizada uma discussão sobre a viabilidade do DETCCS por meio de Prova de Conceito ou *Proof of Concept*, do inglês. Prova de Conceito é uma referência à prova prática de um conceito teórico e descreve a pesquisa nos estágios iniciais, na vanguarda de novas aplicações ou tecnologias e é uma palavra corriqueiramente utilizada para marcar a pesquisa científica como potencialmente extensível e/ou escalável (KENDIG, 2016). Esta prova de conceito foi inspirada no trabalho de BAKHSHI e GHITA (2015) que tinham o objetivo de criar perfis de tráfego de usuário a partir de tendências de uso de aplicativos com base na análise de fluxo de tráfego usando o algoritmo de agrupamento k-means e exploramos sua aplicabilidade a redes definidas por software.

5.2.1 Definição do Objetivo

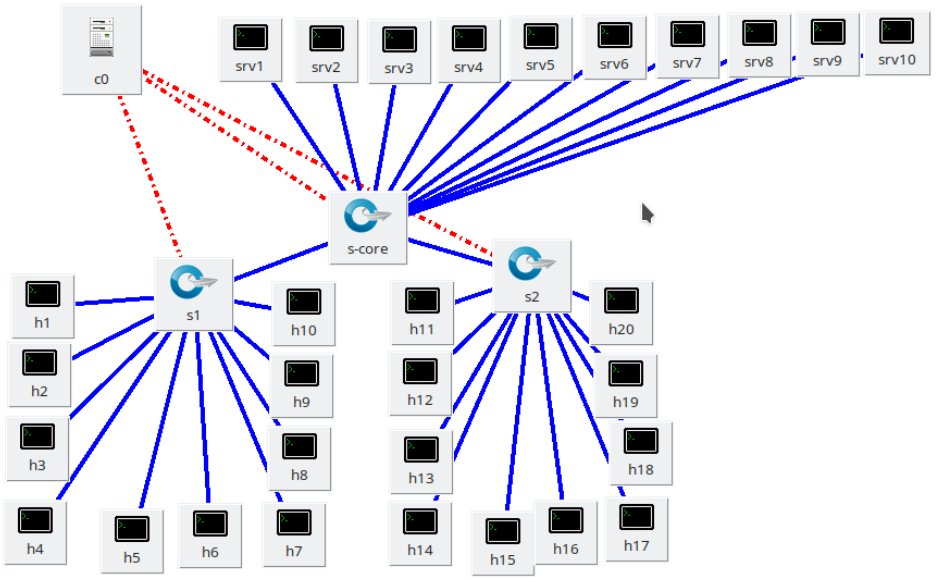
O objetivo de realizar esta prova de conceito é demonstrar, ainda que de forma simples, que a solução, inicialmente teórica, é viável do ponto de vista funcional e, na prática, pode trazer resultados relevantes para administração de redes SDN.

Os conceitos que aqui se desejam provar para o DETCCS são: coleta de estatística de fluxos através do protocolo OpenFlow utilizando o módulo *"StatisticsCollector"* e transmitindo as informações para uma base de dados MongoDB através do Kafka, extração de valor dos dados utilizando algoritmo de aprendizado de máquina e análise exploratória, aplicação de políticas de QoS utilizando OpenFlow a partir do módulo *"QoSManager"*.

5.2.2 Coleta de Estatística

Nessa prova de conceito, utilizamos essa oportunidade para testar o módulo *"Statistics-Collector"*. Para realizar o teste do módulo responsável pela coleta das estatísticas, utilizou-se um cenário simples, que pode ser observado através da figura 29, montado no emulador de rede Mininet (2019). O Cenário foi criado com 1 controlador de rede, 30 hosts e 3 *switchs*, sendo que dos 30 hots, 20 são computadores e 10 são servidores. Os 3 *switchs* operavam com o OPENVSWITCH (2019). O controlador utilizado foi o Floodlight (2019), onde o módulo *"StatisticsCollector"* foi desenvolvido.

Figura 29 – Cenário de Coleta de Dados para Prova de Conceito



Fonte: Autor (2020)

Para realizar a geração de tráfego no cenário em questão, a ferramenta *Distributed Internet Traffic Generator* (D-ITG) foi fundamental e cumpriu bem esse papel. O D-ITG é uma plataforma capaz de produzir tráfego IPv4 e IPv6 ao replicar com precisão a carga de trabalho das aplicações atuais da Internet. Ela também é uma ferramenta de medição de rede capaz de medir as métricas de desempenho mais comuns (por exemplo, taxa de transferência, atraso, *jitter*, perda de pacote) no nível do pacote ((D-ITG, 2019).

O primeiro teste consistiu na geração de 20.000 (vinte mil) comandos no D-ITG para geração de tráfego na rede, 1.000 (mil) para cada host. Os comandos foram gerados aleatoriamente para os destinos (servidores) e contavam com a existência de tráfegos TCP e UDP com diferentes comportamentos e tempo de duração, como por exemplo, tráfego VoIP e game.

Nesse teste, não foram identificadas falhas na coleta de dados pelo módulo desenvolvido, ou seja, todas as informações dos fluxos, incluindo estatísticas, que foram geradas pelo D-ITG e, após inatividade, removidas das tabelas de fluxos dos *switchs*, foram coletadas, transmitidas e armazenadas para uma base de dados. Ainda que seja um teste simples, foi possível validar a

coleta das estatísticas pelo módulo *"StatisticsCollector"* e perceber que era possível seguir nesse caminho.

5.2.3 Extração de Conhecimento

Para demonstrar a extração de conhecimento dos dados estatísticos disponíveis no OpenFlow e mostrar que eles podem auxiliar na administração da rede, sendo alternativas a informações como protocolos, aplicativos ou portas, montamos um cenário cujo o objetivo era ter uma visão sobre os fluxos com base em seu comportamento, inicialmente analisando apenas uma variável simples, que nesse caso foi "tempo de duração do fluxo". Para essa finalidade, uma análise foi desenvolvida no *SparkModule* e utilizou-se o algoritmo de aprendizado de máquina não supervisionado *k-means* para tentar identificar perfis de comportamento. A escolha do k-means foi feita por ele ser um dos algoritmos mais utilizados quando a finalidade é agrupamento e por ter sido utilizado em outros trabalhos, a exemplo de BAKHSHI e GHITA (2015), Amaral et al. (2016) e Jiang et al. (2010).

O primeiro estágio a ser trabalhado foi o de Seleção. Este estágio consiste em criar um conjunto de dados ou focar em um subconjunto, segundo Azevedo e Santos (2008). Desta forma, foram gerados fluxos variando a informação de "tempo de duração"por meio de comandos do D-ITG no ambiente Mininet. O cenário é o mesmo já apresentado na figura 29, mas para esse teste foram utilizados apenas os 10 hosts conectados ao *switch* "s1"e os 10 servidores conectados ao "s-core".

Para no futuro termos embasamento para as discussões a respeito dos resultados da análise, os fluxos gerados precisam ser conhecidos. Partindo dessa ideia, a geração ocorreu da seguinte forma: os 10 hosts foram distribuídos em 3 grupos e receberam IP fixo, conforme mostra a tabela 10. Cada grupo recebeu um percentual de fluxo com uma duração específica, que pode ser visto na tabela 11 e no gráfico 30. Desta forma, é possível perceber que foi gerado propositalmente uma maior quantidade de fluxo com duração curta no grupo 1, longa no grupo 3 e o grupo 2 como intermediário, porém com mais fluxos curtos para embaraçar o cenário.

Tabela 10 – Distribuição dos Hosts em Grupos

Grupo	**Hosts**
Grupo 1	h1,h4,h7,h10
Grupo 2	h2, h5, h8
Grupo 3	h3,h6,h9

Fonte: Autor (2020)

O gráfico 30 foi plotado para tentar mostrar uma visão detalhada acerca de alguns pontos considerados importantes na geração dos fluxos para esse teste. Por exemplo, para o grupo 1 e grupo 2 geramos fluxos com o mesmo percentual de tempo de duração em três intervalos,

Tabela 11 – Percentual de Fluxo Gerado por Grupo

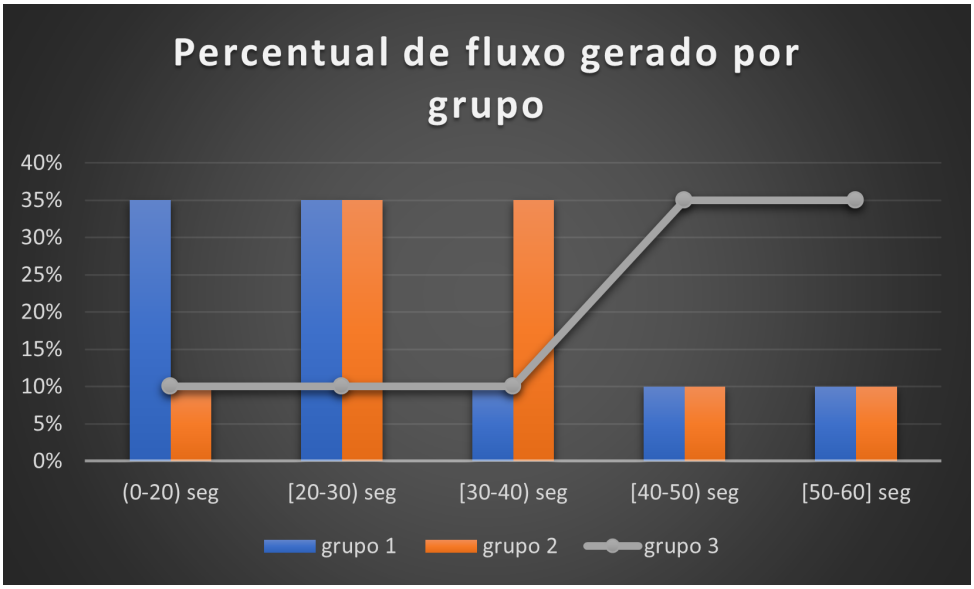
Grupo 1	**Grupo 2**	**Grupo 3**	**Duração min-max**
35%	10%	10%	10-20 seg
35%	35%	10%	20-30 seg
10%	35%	10%	30-40 seg
10%	10%	35%	40-50 seg
10%	10%	35%	50-60 seg

Fonte: Autor (2020)

[20-30]seg;[40-50]seg;[50-60]seg. Já para os grupos 1 e 3, a semelhança acontece em apenas um intervalo, [30-40]seg. Por fim, os grupos 2 e 3 tem semelhança também em apenas um intervalo, que são os fluxos gerados entre [0-20]seg.

.

Figura 30 – Percentual de Fluxo Gerado por Grupo



Fonte: Autor (2020)

Ainda sobre essa etapa, vale lembrar o funcionamento da API desenvolvida para agendamento de tarefas vista na Seção 4.5. Ao desenvolver uma análise no *SparkModule*, é possível configurá-la para receber diversos parâmetros, inclusive parâmetros condicionais para a seleção dos dados.

O módulo *"StatisticsCollector"* faz a captura de todas as estatísticas de fluxos em todos os *switchs* que estejam sob administração do controlador de rede. No entanto, nesse teste desejamos analisar apenas os fluxos do *switch* "s-core", onde passam todos os fluxos da rede. Assim, ao agendar a execução de nossa análise de teste, selecionamos apenas esse *switch*. Seria possível

configurar o período em que os dados foram capturados, por exemplo. Os parâmetros são flexíveis, podem variar entre as análises desenvolvidas.

Após a Seleção, seguiu-se para etapa de Pré-processamento. Esta etapa consiste na limpeza e pré-processamento dos dados para obter dados consistentes segundo Azevedo e Santos (2008). Nesse teste, essa etapa foi de grande relevância, pois foi por meio dela que os dados coletados foram analisados com intuito de remover registros que poderiam prejudicar o resultado da análise. Como o foco era analisar os fluxos que foram gerados propositalmente pelas ferramentas de geração, processo que foi explicado na sessão anterior, foram removidos os registros que possuíam quantidade de pacotes igual a 0, registros da camada de rede diferentes de IPv4 e IPv6, protocolos da camada de transporte diferentes de TCP e UDP.

O Pré-processamento também contribuiu com o módulo *"StatisticsCollector"*, pois, depois de analisar os dados, regras foram inseridas no módulo para evitar que dados inconsistentes fossem enviados para a base.

A etapa de Transformação, segundo Azevedo e Santos (2008), consiste na transformação dos dados usando métodos de redução ou transformação de dimensionalidade. Pois bem, a análise desenvolvida no *SparkModule* utiliza uma matriz de composição de tráfego formada por campos de correspondências do OpenFlow, que são configuráveis no agendamento da análise através da API desenvolvida, e utilizados como entidade escalar, porém, ignorados da perspectiva de agrupamento pelo algoritmo. Essa prova de conceito foi baseada nos trabalho de BAKHSHI e GHITA (2015) e Jiang et al. (2010) e utilizou-se apenas do IP de origem do fluxo como entidade escalar, mas seria possível combinar com o IP de destino, porta de origem e destino e protocolo, etc. As dimensões do vetor, essas utilizadas para fins de agrupamento pelo k-means, nesse teste são formadas por uma heurística que divide o fluxo em quatro dimensões pelo tempo de duração do fluxo, d1=(0,15 seg), d2=[15,30 seg), d3=[30,45 seg), d4=[45 seg + ∞). A ideia da análise é distinguir os fluxos pelo tempo de duração e apontar indícios de comportamento elefantes e ratos, por exemplo. O tempo de duração sozinho não é suficiente para fazer essa caracterização mas é o suficiente para realizar a prova de conceito. A matriz de composição ilustrada na tabela 12 exemplifica bem o texto.

Tabela 12 – Exemplo de uma Matriz de Composição de Duração do Fluxo

ip_src	**qtd.fluxo**	**d1**	**d2**	**d3**	**d4**
[192.168.1.1]	70	37	23	7	3
[192.168.1.2]	80	5	10	30	35
[192.168.1.3]	150	90	40	15	5
[192.168.1.4]	30	0	8	10	12
...	...	...	...	...	...

Como todas as dimensões que serão processadas pelo algoritmo de aprendizado de máquina não supervisionado estão em uma mesma unidade de medida, não foi aplicado nenhum

método de normalização para esse exemplo.

A fase de *Data Mining*, como visto no início desse capítulo, consiste na busca de padrões de interesse em uma determinada forma representacional, dependendo do objetivo de mineração de dados (AZEVEDO; SANTOS, 2008). Desta forma, buscamos encontrar padrões de tráfego analisando a duração do fluxo, utilizando o algoritmo k-means. Conforme visto no código 2, alguns parâmetros precisam ser definidos para o agendamento e execução de uma análise. Na análise em questão, que utiliza o algoritmo *k-means*, um parâmetro importante é *k*, que representa o valor escolhido para o número de *clusters* exclusivos que se deseja obter. Para essa prova de conceito, o *k* foi escolhido por observação, mas existem métodos (Cotovelo, Coeficiente da Silhueta, etc) que serão implementados na solução para obter um bom valor *k* para o *k-means*.

A análise foi executada com *k*=4 e utilizou o tráfego do "s-core"nos dois sentidos. O resultado do agrupamento feito pelo *k-means* considerando o tempo de duração dos fluxos pode ser vista na tabela 13.

Tabela 13 – Resultado de Agrupamento Realizado pelo K-means

Grupo	**Hosts**
Cluster 1	h1,h2,h4,h5,h7,h8,h10
Cluster 2	h3,h6,h9
Cluster 3	srv3,srv6,srv9
Cluster 4	srv1,srv2,srv4,srv5,srv7,srv8,srv10

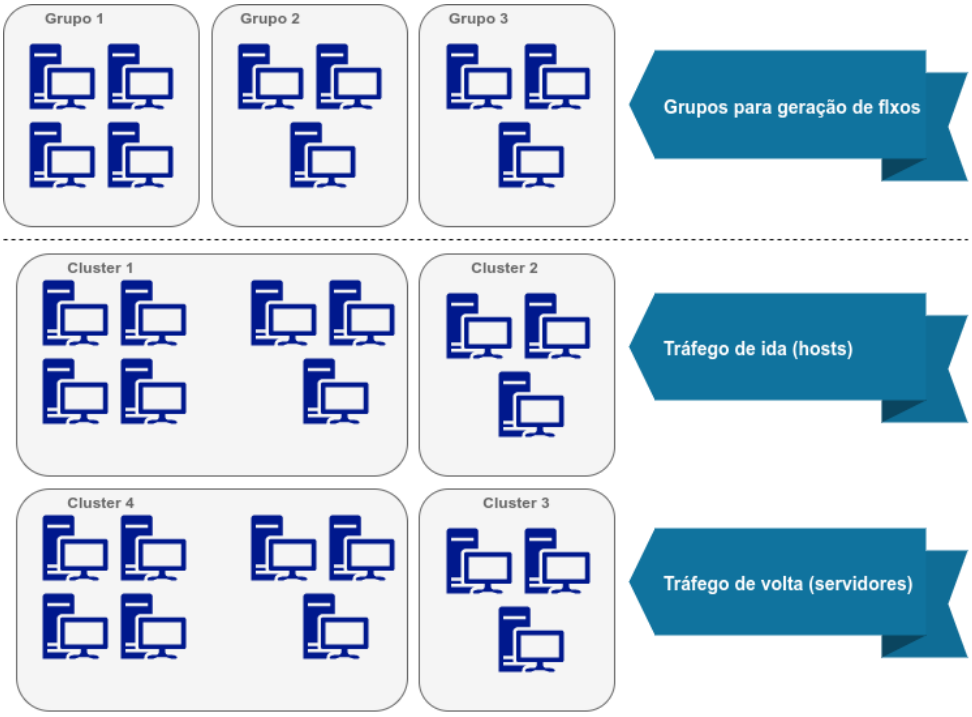
Fonte: Autor (2020)

Em relação a Interpretação, a observação dos 4 *clusters* resultantes após a execução do k-means, vistos na tabela 13, permite a percepção que o *cluster* 1 foi formado pelos hosts do grupo 1 e grupo 2 e que o *cluster* 2 foi formado exclusivamente pelos hosts do grupo 3. Os *clusters* 3 e 4, curiosamente foram formados pelos fluxos de volta, ou seja, quando os servidores originavam os fluxos. O *cluster* 3 com os servidores que retornaram fluxo para os hosts do grupo 3 e o *cluster* 4 formado pelos servidores que enviaram fluxos para os hosts dos grupos 1 e 2. A figura 31 ilustra o resultado e tenta facilitar a compreensão do texto.

O primeiro olhar para o resultado da clusterização foi satisfatório, pois, foi possível notar que o k-means conseguiu criar *clusters* consistentes analisando o comportamento dos fluxos (tempo de duração), compatíveis com o comportamento provocado propositalmente no experimento.

Na geração do tráfego, 70% do fluxo gerado pelos hosts do grupo 3 eram considerados longos em nosso experimento, eles variavam entre 40 e 60 seg. Nos demais grupos, esse tráfego era de apenas 10%. Após execução da análise, esse comportamento foi identificado pelo algoritmo de agrupamento que criou dois *clusters*, *clusters* 2 e *clusters* 3. Um para o fluxo de ida e outro para o fluxo de volta. Já os fluxos de ida dos grupo 1 e 2 foram unificados no *cluster* 1 e os fluxos

Figura 31 – Resultado de Agrupamento Realizado pelo K-means



Fonte: Autor (2020)

de volta foram unificados no *cluster* 4.

Esse simples resultado foi aceito como satisfatório para a prova de conceito, tendo em vista que, para essa análise, o número de *clusters* foi fixado em 4 e o tráfego foi analisado nos dois sentidos. O resultado condiz com o esperado, visto que foi analisado o comportamento do tráfego observando apenas o tempo de duração dos fluxos. Neste sentido, é coerente o grupo 1 e grupo 2 pertencerem a um único *cluster*, visto que eles tinham semelhança em 3 intervalos na geração de fluxo, enquanto o grupo 3 tinha apenas 1 intervalo de semelhança com os demais. Desta forma, foi possível aceitar o algoritmo de agrupamento k-means seguir nas próximas etapas da pesquisa.

5.2.4 Regras de QoS

O resultado da análise foi a evidência de 4 *clusters*. Esses *clusters* serão rotulados como classes para obterem tratamentos exclusivos nas regras de QoS. Eles foram formados por hosts que tinham comportamentos semelhantes mediante a variável analisada, no caso, o tempo de duração do fluxo. Essa classificação torna possível obter outras informações relevantes para tomada de decisão na administração da rede, como por exemplo, o volume trafegado em cada *cluster*; a quantidade de hosts em cada *cluster*, etc.

Cabe ao administrador da rede definir as regras de QoS apropriadas de acordo com os resultados analisados e associar essas regras aos grupos evidenciados pela análise. Essa prova de

conceito utilizou as regras vistas na tabela 14. Inicialmente essas regras possuem características simples, a exemplo da limitação de taxa de transferência. Essas regras serão aplicadas por meio de medidores do *switch OpenFlow*. Quando as regras são definidas, o controlador SDN configura entradas da tabela de fluxo para cada conexão com base em seu número de classificação para instruir como os fluxos serão executados. Logo após, um medidor é conectado diretamente às entradas de fluxo para medir e controlar a taxa de dados dos pacotes.

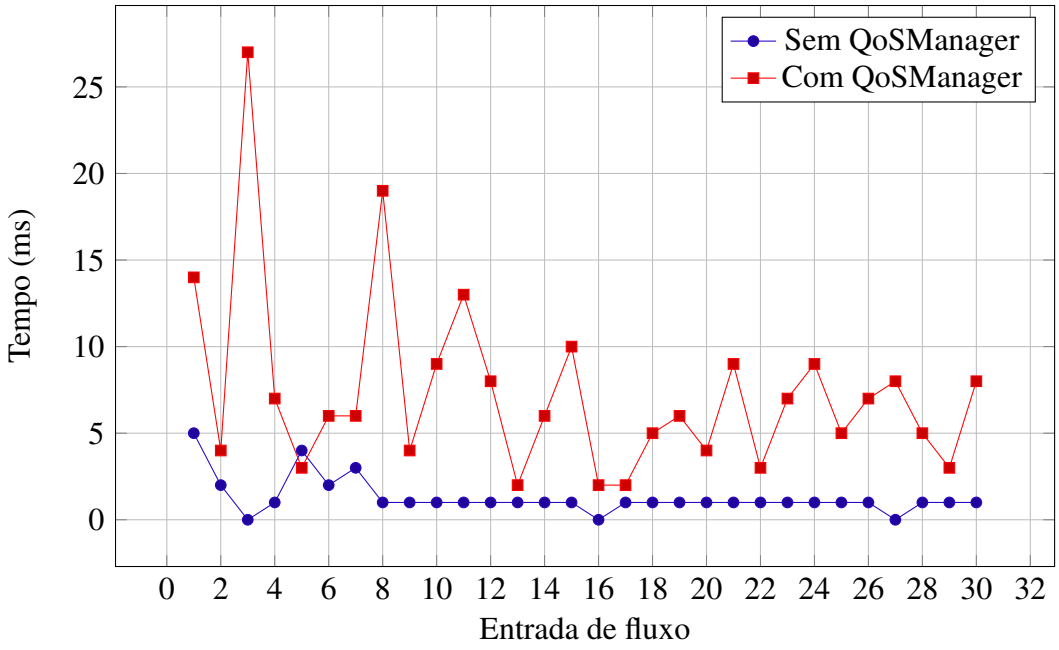
Tabela 14 – *Clusters* Associado a uma Fatia da Rede

cluster	**host_src**	**taxa(Mbits/s)**
Cluster 1	h1,h2,h4,h5,h7,h8,h10	2Mbits/s
Cluster 2	h3,h6,h9	3Mbits/s
Cluster 3	srv3,srv6,srv9	3Mbits/s
Cluster 4	srv1,srv2,srv4,srv5,srv7,srv8,srv10	2Mbits/s

Fonte: Autor (2020)

A rotina de verificação para aplicar ou alterar um medidor no *switch OpenFlow* é configurável pelo administrador nos parâmetros do módulo *QoSManager*; nesse teste a configuração foi de 10 segundos. Já o tempo necessário para criar uma entrada de fluxo associada a um medidor OpenFlow poderá ser visto no gráfico 32. A medição desse tempo é iniciada quando o controlador é notificado com a chegada de uma mensagem assíncrona de entrada de pacote (*Packet-In*) até o momento que o controlador envia para o *switch* uma entrada para uma tabela de fluxo. Quando o *QoSManager* estiver ativo, ele irá consultar a existência de regras de QoS correspondente ao fluxo, essa comparação é realizada através dos campos de correspondência do OpenFlow definidos pelo administrador. No caso desse teste, o IP de origem é o campo de correspondência a ser comparado. Se o novo fluxo corresponder a alguma regra, o *QoSManager* fará a associação da entrada de fluxo com o medidor para limitar a taxa de transferência. O gráfico da figura 32 mostra o desempenho do controlador Floodlight ao criar entradas de 30 fluxos no switch "s-core"do cenário mostrado na figura 29, com a presença e sem a presença do *QoSManager*.

Figura 32 – Tempo para Criar Entrada de Fluxo



Fonte: Autor(2020)

É possível observar que o *QoSManager* causa um pequeno atraso de milissegundos na criação da entrada de fluxo no *switch*, em média, 4,25ms. No entanto, é aceitável visto que isso acontece apenas no momento da criação da entrada de fluxo, quando o controlador é consultado sobre qual ação tomar. O trabalho de (LE et al., 2018) apresenta delays de até 5s, mas é importante frisar que nesse trabalho são analisados lotes de dados e são aplicados modelos de aprendizado supervisionado para classificar o fluxo.

O teste da limitação de taxa de transferência usando o medidor OpenFlow foi realizado com o Iperf, que é um software utilizado para gerar tráfego e pode ser operado para testar a largura de banda. Nesse sentido, o host h1, classificado no *cluster* 1, visto na tabela 14, teria um medidor do tipo *ofp_meter_band_drop* no switch "s-core", limitando a taxa de transferência a 2 Mbits/s. Os comandos vistos no código 3 do iperf é bidirecional a uma largura de banda de 3 Mbits/s, ou seja, superior a limitação imposta pelo medidor.

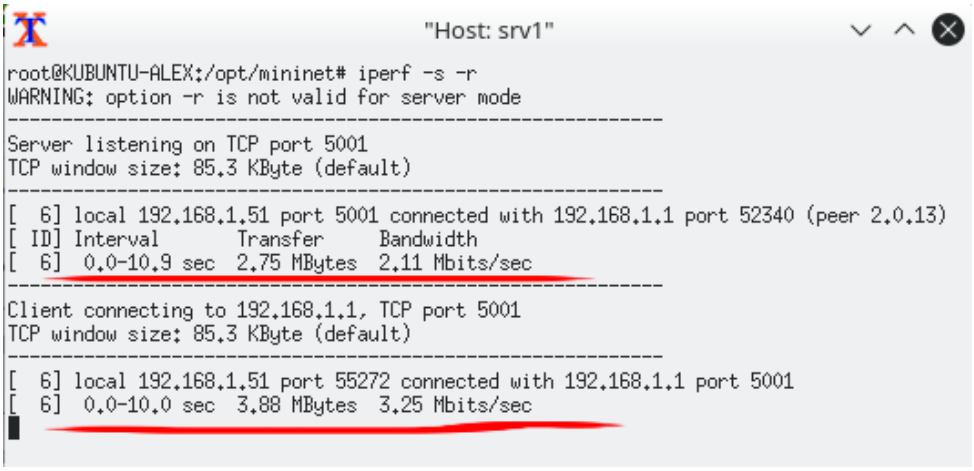
Código 3 – Comando Iperf

```
iperf -c 192.168.1.51 -r -b 3M -f m  #Comendo no Cliente
#-c host de destino
#-r teste bidirecional individualmente
#-b largura de banda
#-f formato do relatório em: Mbits.
iperf -s -r #Comando no Servidor
```

Fonte: Autor (2020)

No primeiro teste, aplicou-se apenas a primeira regra da tabela 14. Assim, a regra seria aplicada nos fluxos em que a origem fossem h1,h2,h4,h5,h7,h8 ou h10. Nesse teste, os hosts utilizaram IP fixo, mas seria possível utilizar o endereço físico, *MAC Address* ou até mesmo uma combinação de campos de correspondência do OpenFlow, conforme mostra o código 1. A figura 33 mostra o relatório do Iperf após a conclusão do teste. É possível perceber na primeira linha grifada de vermelho que a limitação da taxa de transferência foi aplicada quando o fluxo partia do h1 para srv1, mas quando o srv1 é a origem não existiu limitação de taxa de transferência.

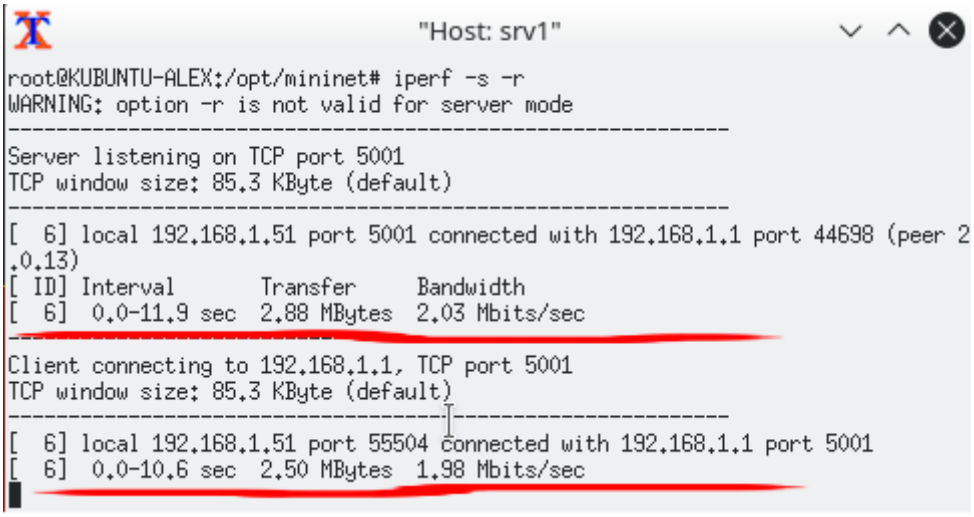
Figura 33 – Taxa de Transferência com Medidor OpenFlow em Única Direção



Fonte: Autor (2020)

No segundo teste, aplicou-se a primeira e quarta regra da tabela 14. Desta feita, a limitação da taxa foi aplicada em ambas as direções, conforme mostra figura 34.

Figura 34 – Taxa de Transferência com Medidor OpenFlow em Dupla Direção



Fonte: Autor (2020)

5.3 Execução do Estudo de Caso

Esta Seção apresenta um estudo de caso que utiliza dados extraídos de uma rede real. As subseções seguintes definem o objetivo do estudo de caso, bem como o processo de extração de conhecimento e aplicação de políticas de QoS.

5.3.1 Definição do Objetivo

O objetivo deste experimento é avaliar, por meio de um estudo de caso, o uso do DETCCS para coletar informações dos fluxos dos dispositivos de uma SDN e a utilização de estruturas de *big data* para analisar e compreender os padrões de tráfego de rede, identificar classes de tráfego, e aplicar politicas que contribuam com a garantia na qualidade do serviço em um ambiente emulado com a ferramenta Mininet.

Da mesma forma que foi feito na primeira prova de conceito, será necessário o desenvolvimento de uma nova análise no *SparkModule* e essa análise irá montar um vetor composto com entidade escalar e dimensões de características para que o algoritmo de aprendizado de máquina não supervisionado k-means seja executado e consiga, através das características analisadas, agrupar o fluxo com comportamento semelhante. Segundo Silva et al. (2015), a identificação de tráfego pode ser realizada por agrupamento de acordo com o comportamento dos fluxos de dados na rede e os fluxos podem ser classificados por classes de dados realizando uma analogia com as características de alguns animais. Por exemplo, fluxos pequenos como “ratos”, fluxos grandes como “elefantes”, fluxos rápidos como “libélulas” e fluxos lentos como “tartarugas”.

Nessa etapa, a seleção de características tem grande importância e é fundamental para o bom funcionamento do algoritmo de agrupamento. A coleta de estatística por fluxo do OpenFlow fornece poucas características, no entanto Amaral et al. (2016) e Kuranage, Piamrat e Hamma (2019) se basearam apenas nessas informações, que são nativas do OpenFlow, para criar grupos de comportamento de tráfego.

5.3.2 Extração de Conhecimento

Na prova de conceito foi usado um conjunto de dados gerados por meio de ferramentas específicas para esse propósito. Além da geração, testamos a captura desses dados com o módulo *"StatisticsCollector"* desenvolvido para o controlador Floodlight. No entanto, para esta etapa do trabalho, o conjunto de dados *"IP Network Traffic Flows Labeled with 75 Apps"* do banco de dados Kaggle foi utilizado. Este conjunto de dados foi uma boa combinação para satisfazer os objetivo do trabalho, pois ele possui três componentes importantes de um bom conjunto de dados, que são: do mundo real, substancial e diverso. Este conjunto de dados foi criado por meio de coleta de dados da rede da *Universidad Del Cauca*, que fica localizada em *Popayán* na Colômbia durante seis dias (26, 27, 28 de abril e 9, 11 e 15 de maio) de 2017. Para capturar os dados foram usadas várias ferramentas de captura de pacote e ferramentas de extração de

dados. Ele consiste em 3.577.296 instâncias e 87 características originalmente projetadas para classificação de aplicativos. Mas, para esse trabalho, apenas uma fração dessas características do conjunto de dados foram utilizadas. Cada linha representa um fluxo de tráfego de uma origem a um destino e cada coluna representa características do tráfego de dados (ROJAS, 2019).

Na preparação dos dados, apenas algumas características foram selecionadas. O critério utilizado foi selecionar apenas as características relacionadas ao objetivo do trabalho, que poderiam ser facilmente obtidas pelo controlador de rede através do módulo *"StatisticsCollector"* sem a necessidade de outros protocolos, ferramentas ou aplicativos de terceiros. Assim, informações como *MAC Address* de origem e destino, portas, endereços IPs, duração do fluxo, quantidade de bytes de fluxo, quantidade de pacotes e tamanho médio do pacote foram preservados, as demais informações foram removidas do conjunto de dados. Os detalhes dessas informações podem ser vistos na Seção 4.2, onde tem um detalhamento acerca do coletor de estatísticas.

Nessa etapa de Pré-processamento, os esforços foram aplicados para remover eventuais registros inconsistentes e que poderiam prejudicar o resultado da análise. Por exemplo, registros em que as características alvo de análise estivessem com valor 0 poderiam ser prejudiciais e por esse motivo foram removidos. Ao final desa etapa, 546.810,00 registros foram removidos do conjunto de dados, restando 3.030.486,00 para serem analisados.

Em relação a Transformação dos dados para esse estudo de caso, os fluxos foram analisados por uma perspectiva de agrupamento, tendo como entidade escalar cinco características do fluxo, que são: IP de origem e destino, porta de origem e destino e protocolo. Para formar as dimensões do vetor e possibilitar a execução do algoritmo k-means foram utilizadas as informações de estatísticas de fluxos fornecidas pelo protocolo OpenFlow, que são: quantidade de bytes e pacotes e duração do fluxo. A partir dessas informações e inspiração no trabalho de Silva et al. (2015), foi possível estimar outras características estatísticas e escalares, como por exemplo a quantidade de bytes por segundos, a quantidade de pacotes por segundos, o mínimo, máximo, média e desvio padrão de pacote e bytes por fluxo.

Os dados coletados têm características distintas com valores em escalas diferentes. A fim de evitar que recursos com valores de grande escala se sobreponham aos de menor escala, causando mau desempenho no algoritmo, os dados coletados são normalizados para que fiquem dentro de um padrão que facilite a sua análise. Assim como em Amaral et al. (2016), Kuranage, Piamrat e Hamma (2019), a normalização Min-Max foi utilizada para esse fim. Min-Max é um processo de transformar valores discrepantes usando uma escala que vai de 0,0 (para o menor) e 1,0 para o maior valor. A fórmula se dá pela seguinte equação:

$$z = \frac{x - min(x)}{max(x) - min(x)} \tag{5.1}$$

Com os dados normalizados, a análise desenvolvida no *SparkModule* segue para etapa de *Data Mining*. Para essa etapa, foi utilizado o algoritmo de aprendizado de máquina não

supervisionado k-means. O uso de técnicas de aprendizado de máquina para classificar tráfego já foram utilizadas em diversos trabalhos (BAKHSHI; GHITA, 2015; AMARAL et al., 2016; JIANG et al., 2010) e podem ser vistas em muitos pontos dessa dissertação. Sobre esse assunto especificamente e sobre o algoritmo k-means a Subseção 2.6.2 tenta fazer um apanhado mais detalhado.

Um dos desafios da utilização do k-means é saber qual é o número de *clusters* ideal para um conjunto de dados. É possível encontrar na literatura algumas técnicas para vencer esse desafio, por exemplo, em Syakur et al. (2018) o Método Cotovelo (do inglês *Elbow Method*) é utilizado. Já Kuranage, Piamrat e Hamma (2019) usa a técnica *Davies-Bouldin*. Nesse trabalho, a análise desenvolvida no *SparkModule* utilizou o método conhecido por Coeficiente da Silhueta (LEARN, 2020).

A análise de silhueta pode ser usada para estudar a distância de separação entre os *clusters* resultantes. O gráfico de silhueta exibe uma medida de quão próximo cada ponto em um *clusters* está de pontos nos *clusters* vizinhos e, portanto, fornece uma maneira de avaliar parâmetros como o número de *clusters* visualmente. Esta medida tem um intervalo entre [-1, 1] (LEARN, 2020).

Coeficientes de silhueta próximos a 1 indicam que a amostra está longe dos *clusters* vizinhos. Um valor de 0 indica que a amostra está no ou muito próximo do limite de decisão entre dois *clusters* vizinhos. Os valores negativos indicam que essas amostras podem ter sido atribuídas ao *clusters* errado (LEARN, 2020).

A fórmula para obter o coeficiente de silhueta se dá pela seguinte equação (COMMUNITY, 2020):

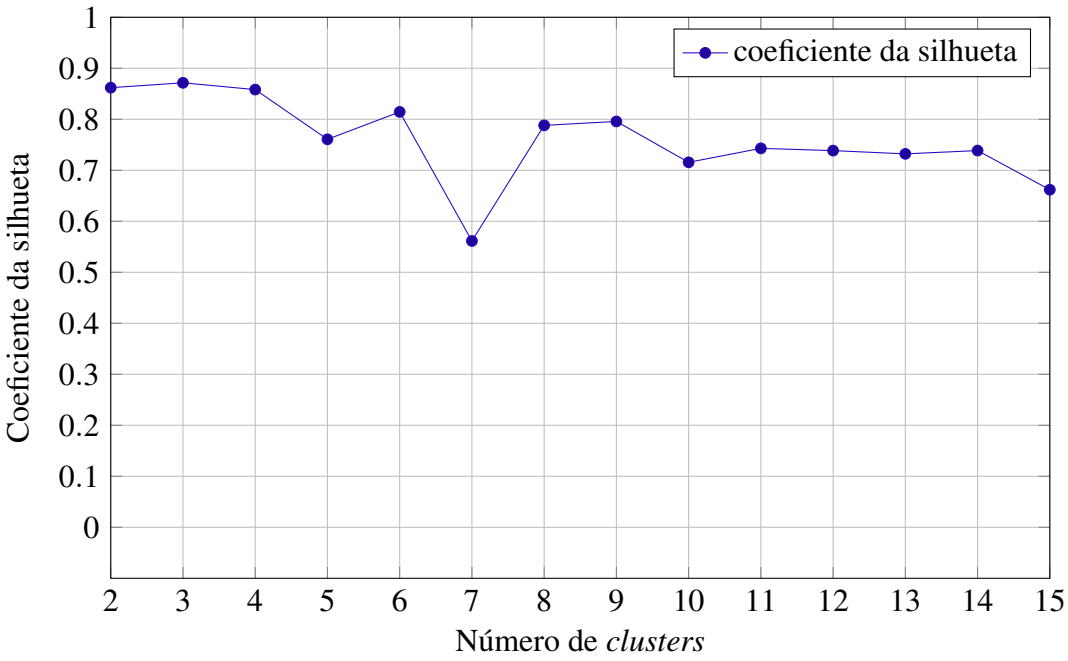
$$s_i = \frac{b_i - a_i}{max(a_i, b_i)} \tag{5.2}$$

Para cada objeto i obtêm-se o valor s_i, onde:

- a_i = é a dissimilaridade média do objeto i em relação a todos os outros objetos do seu *clusters*.
- b_i = é a dissimilaridade média do objeto i em relação a todos os outros objetos do *clusters* vizinho mais próximo.

Desta forma, o k-means foi executado 14 vezes, onde o número de *clusters* sofreu uma variação entre 2 e 15. Em cada rodada da execução, o coeficiente da silhueta era calculado e armazenado. O *cluster* com a maior pontuação, ou seja, mais próximo de 1, foi o escolhido.

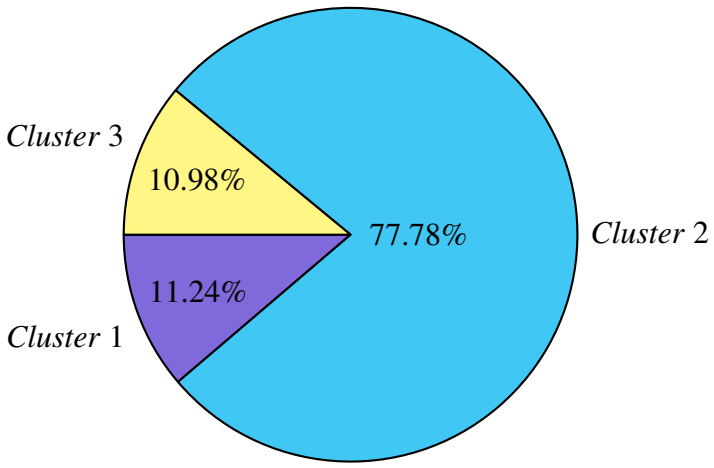
Figura 35 – Coeficiente da Silhueta para Escolha do Número de *Clusters*



Fonte: Autor(2020)

Em relação a Interpretação, observa-se que, baseado nos resultados obtidos com o teste dos coeficientes de silhueta, o comportamento do conjunto de dados é bem representado por 3 *clusters*, mediante as características analisadas. O primeiro ponto verificado foi a distribuição dos fluxos em cada *cluster*. Analisando o gráfico ilustrado na figura 36, observa-se uma grande concentração de fluxo no *cluster 2* e um equilíbrio entre o *cluster* 1 e 3.

Figura 36 – Percentual de Distribuição de Fluxos por *Clusters*

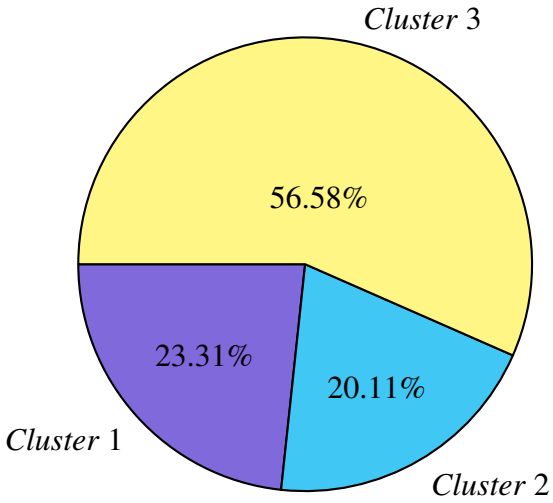


Fonte: Autor(2020)

Apesar do *cluster* 2 possuir maior quantidade de fluxos, ele se mostra o menor grupo em transmissão de pacotes. Nessa característica, os *clusters* 1 e 2 são bastante parecidos, enquanto o

3 se mostra um grupo diferenciado nesse quesito, chegando a ser maior transmissor de pacotes que todos os outros grupos juntos. Esses dados são representados em percentuais no gráfico de pizza da figura 37. Esse desempenho dá indícios que o *cluster* 3 possui um comportamento de fluxos elefantes e o *cluster* 2 comportamento de fluxos ratos. No entanto, para chegar a essa conclusão, será necessário analisar mais duas características do grupo: o volume e a duração dos fluxos.

Figura 37 – Percentual de Pacotes Transmitidos por *Cluster*

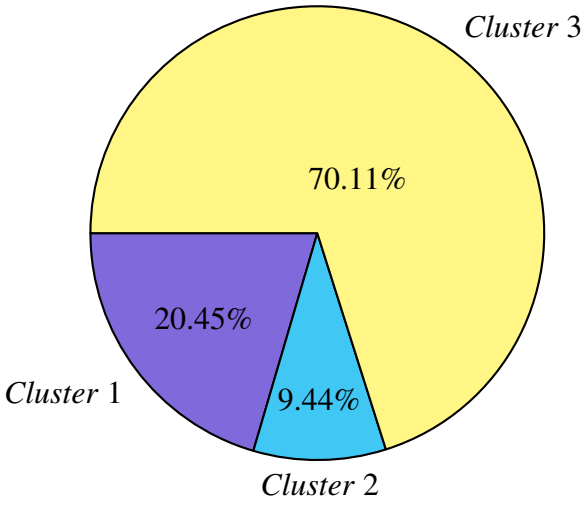


Fonte: Autor(2020)

No gráfico 38 foi plotado em percentual o volume de dados que cada *cluster* transmitiu. O que se pode observar é um comportamento semelhante ao do gráfico de percentual de pacotes transmitidos, o 37, onde o *cluster* 3 sobressai sobre os demais. Aqui, ele atinge a marca de 70.11% do volume transmitido na rede. Para um administrador de rede, essa é uma informação relevante, visto que os fluxos maiores podem impactar significativamente no trafego de fluxos menores (SILVA et al., 2018).

Seguindo a mesma linha, o *cluster* 2 foi o grupo com menor percentual em termos de volume, com apenas 9.44% do total de volume trafegado. Isso aumenta consideravelmente as possibilidades desse grupo ser tratado por administradores de rede como sendo fluxos ratos. Vilela (2006) afirma que os fluxos elefantes são responsáveis pelo maior volume de bytes na rede, no entanto, a quantidade de fluxos muito pequenos (ratos) é muito maior.

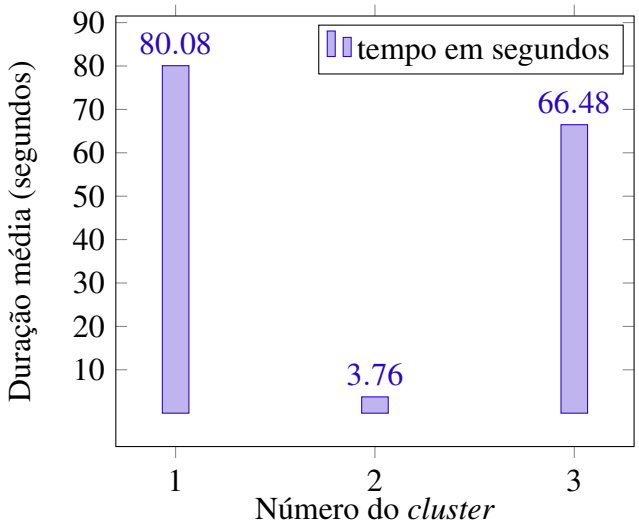
Figura 38 – Distribuição do Volume nos *Clusters*



Fonte: Autor(2020)

Por fim, a duração média dos fluxos de cada *cluster* foi ilustrada no gráfico da figura 39. Nesse gráfico, é possível perceber que o *cluster* 3 tem uma duração considerada longa no contexto do cenário em questão. Assim, com base no que diz Silva et al. (2018), concluímos que esse *cluster* é formado por fluxos que demonstram ter o comportamento elefante. Para Silva et al. (2018), os fluxos elefantes tendem a ter uma duração relativamente alta e apresentam a maior parte do volume do trafego da infraestrutura, apesar de tenderem a ser uma fatia pequena dentre todos os fluxos. O trabalho de Rocha et al. (2017) chega a afirmar que os fluxos elefantes representam apenas 10% dos fluxos e trafegam a maior parte do volume de dados, transmitindo até mais de 80% da quantidade de bytes de toda a rede. O cenário descrito por Rocha et al. (2017) é bem semelhante com o que nosso mecanismo encontrou.

Figura 39 – Duração Média de Fluxos nos *Clusters*

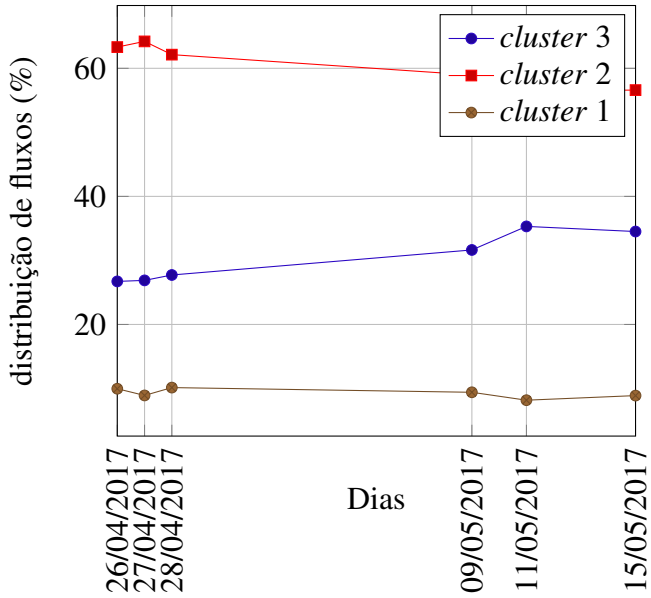


Fonte: Autor (2020)

A análise desenvolvida no *SparkModule* pode contemplar uma análise exploratória dos *clusters*, procurando responder algumas perguntas cujas respostas podem ajudar aos administradores a conhecer melhor o cenário da rede. Por exemplo, questionamentos como, quais os hosts de cada *cluster* que mais enviam pacotes? Quais os que trafegam maior volume de bytes na rede? Qual a porta de destino mais utilizada? Qual horário ou intervalo do dia que mais se trafega dados? Qual o protocolo da camada de transporte mais utilizado? Enfim, estas perguntas ficam abertas para o desenvolvimento de cada análise.

Por exemplo, no gráfico da figura 40 foi analisado a distribuição de fluxos nos *clusters* por meio de uma visão temporal e foi observado um comportamento com pequenas variações durante os dias analisados. No entanto, em um cenário real isso pode sofrer grandes variações e o administrador pode recalcular os *clusters* assim que perceber. Também seria possível implementar um rotina para recalcular os *clusters* caso o comportamento do tráfego divergisse em uma margem determinada pelo administrador, mas o tempo da pesquisa não permitiu a implementação dessa funcionalidade na prática.

Figura 40 – Percentual de Distribuição de Fluxos por *Clusters* por Dia



Fonte: Autor(2020)

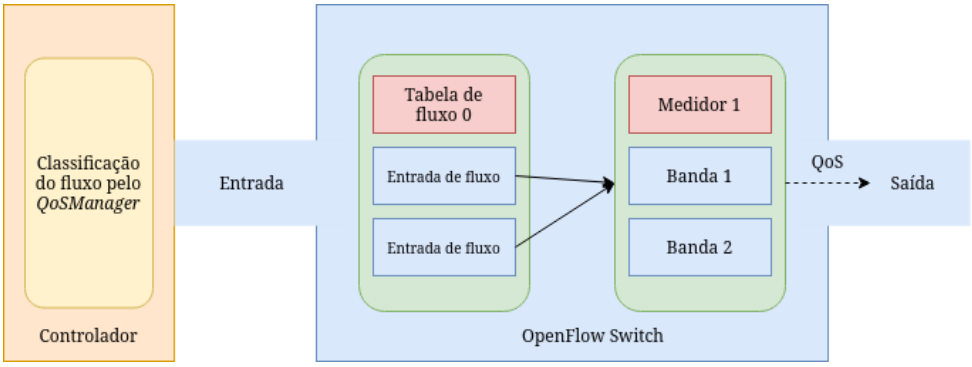
Na prática, o que foi implementado foi permitir que a análise seja agendada para recalcular os *clusters* em períodos definidos pelo administrador através da API desenvolvida ou que os fluxos sejam analisados por uma combinação diferente de características. Desta forma, o administrador estará sempre atualizado sobre o cenário da rede para tomar as melhores decisões.

5.3.3 Divisão da Rede Baseado na Classificação

Depois que o número de *cluster* mais adequado for escolhido no conjunto de dados e o algoritmo de agrupamento for executado, a análise desenvolvida no *SparkModule* armazena os resultados no banco de dados para que sejam disponibilizados ao administrador da rede. O objetivo é que os resultados tragam informações relevantes sobre os *clusters* ao administrador para que sejam tomadas decisões importantes em relação aos recursos da rede.

Devido à compatibilidade com Open vSwitch e ao curto tempo dessa pesquisa, só foi possível prover a possibilidade de aplicar aos fluxos limitação na taxa de transferência usando tabelas de medidores do OpenFlow. Um medidor mede a taxa de pacotes atribuídos a ele e permite controlá-las. Os medidores são conectados diretamente às entradas de fluxo, conforme demonstra a figura 41 (FOUNDATION, 2012).

Figura 41 – Fatiamento de Rede com Base na Classificação de Tráfego



Fonte: Autor (2020)

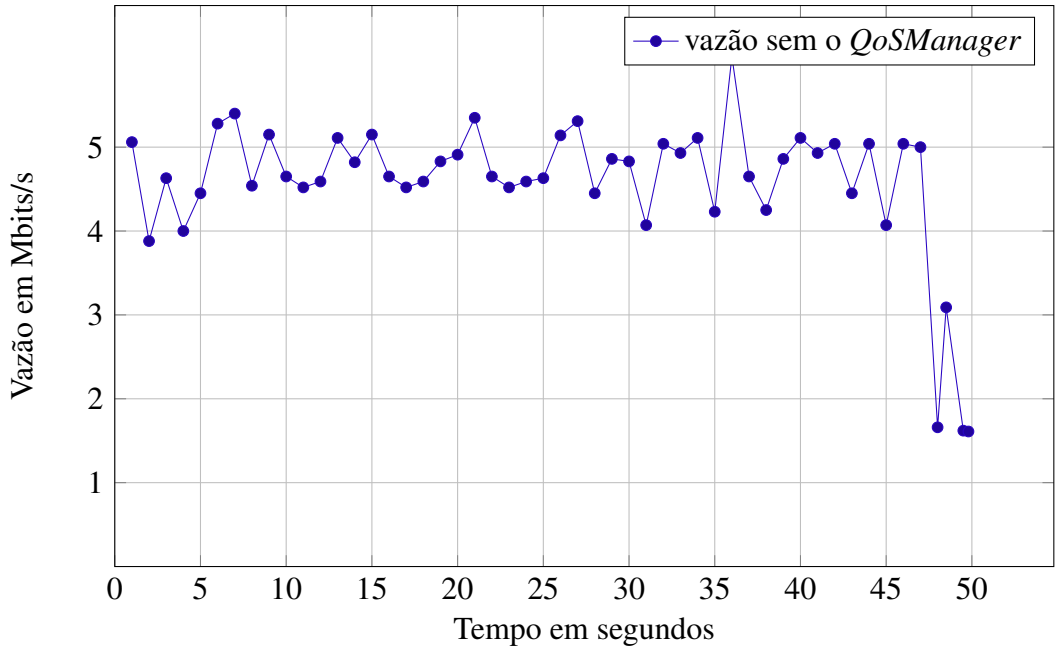
Desta forma, o administrador pode associar um *cluster* resultante da análise desenvolvida no *QoSManager* a um medidor e limitar a taxa de transferência. Assim, o *QoSManager* consulta e verifica através dos campos de correspondência se um novo fluxo corresponde a alguma regra definida pelo administrador e instrui como os fluxos (pacotes) são executados. Logo após, um medidor é conectado diretamente às entradas de fluxo para medir e controlar a taxa de dados dos pacotes. Como pode ser visto na imagem 41, em cada tabela podem existir várias entradas de medidores, que são definidas para tráfegos específicos. Uma vez escolhida a banda, o medidor aplica o processamento de QoS aos pacotes do fluxo e segue para o próximo ponto da rede ou ao destino final.

Para esse teste, o mesmo cenário hipotético visto na imagem 29 foi utilizado. Apenas três computadores(h1,h2,h3) e um servidor(srv1) foram utilizados. Cada computador pertence a um *cluster* diferente. O link entre o *switch* "s-core"e o servidor 1(srv1) foi limitado a uma largura de banda de 5 Mbit/s. A ferramenta Iperf foi utilizada para gerar os fluxos e testar o cenário com presença e sem a presença do *QoSManager*.

O primeiro teste foi a geração de 3 conexões simultâneas sem a presença do *QoSManager*. Cada conexão de h1 para srv1 envia um aquivo equivalente a 10 Mbytes. O segundo teste foi aplicação do mesmo envio, só que dessa vez o *QoSManager* estava limitando os fluxos do *cluster 1* a 2,5 Mbits/s. Os gráficos 42 e 43 representam esse teste inicial.

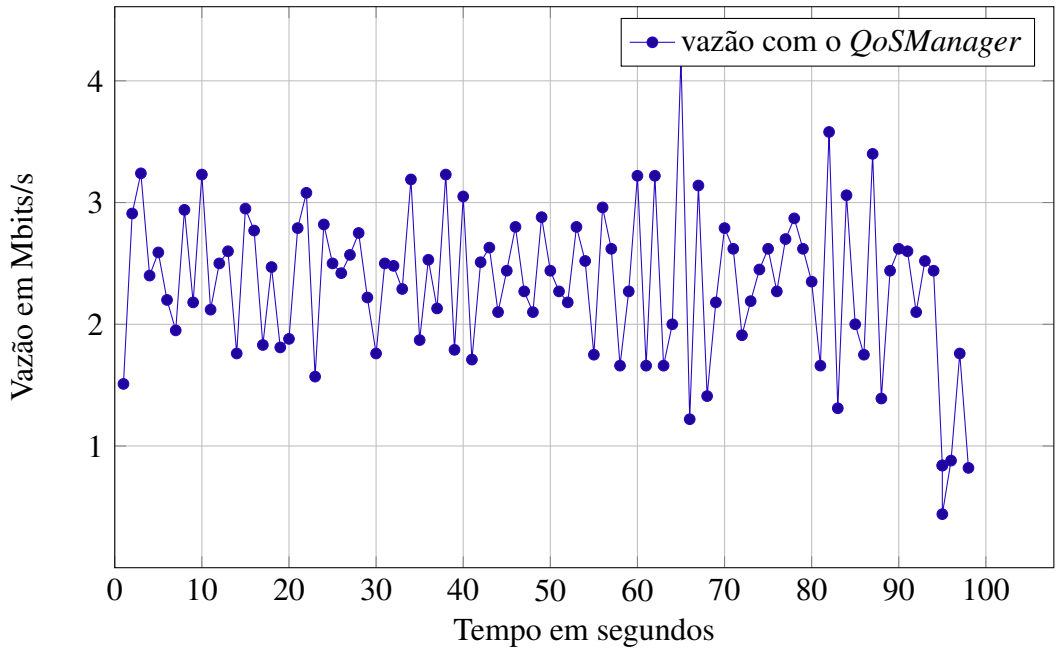
A figura 42 mostra um cenário em que a banda reservada para o link (5 Mbits/s) é totalmente consumida pelos tráfegos gerados por h1. Já o gráfico 43, mostra que apenas metade do link (2,5 Mbits/s) foi utilizado. Em contrapartida, é possível perceber através dos gráficos, que o tempo necessário para transferir o mesmo arquivo dobrou. Assim, esse teste serviu para demonstrar que é possível definir o comportamento de um *cluster* associando uma determinada taxa de transferência a ele, utilizando apenas tabela de medidores do OpenFlow.

Figura 42 – Vazão sem o *QoSManager*



Fonte: Autor(2020)

Figura 43 – Vazão com o *QoSManager*



Fonte: Autor(2020)

6
Conclusão

A conclusão desta dissertação é apresentada nesse capítulo. Além da conclusão, são tratadas as dificuldades e limitações enfrentadas para o desenvolvimento do trabalho proposto até aqui. Também são abordados pontos que podem originar trabalhos futuros, dando continuidade a esta pesquisa.

A Rede Definida por Software traz uma nova abordagem para o contexto das redes de computadores ao separar o plano de dados do plano de controle. Isso garante à SDN o controle centralizado e torna a tarefa de obter informações de tráfego e falhas da rede mais simples com essa visão globalizada. Combiná-la com tecnologias de *big data* e aplicar técnicas de aprendizado de máquina para extrair conhecimento da rede pode trazer benefícios notáveis para diversas áreas, a exemplo da engenharia de tráfego.

Desta forma, como já mencionado, esse trabalho tem o objetivo de prover um mecanismo de auxílio à tomada de decisões sobre o tráfego em SDN, baseado em: coleta de informações dos fluxos de dados, estruturas de *big data*, identificação de padrões e classes de tráfego e aplicação de políticas para garantir QoS ou maximizar o aproveitamento da rede.

Para alcançar o objetivo geral, uma metodologia de pesquisa foi definida e seguir o planejado em cada etapa estabelecida foi de suma importância para a conclusão do trabalho. Pode-se observar que o objetivo final foi parcialmente alcançado, por meio da junção dos objetivos específicos. Parcialmente porque julga-se necessário realizar testes mais intensos e em cenários mais complexos que serão apresentados na sessão de trabalhos futuros.

A partir do objetivo de desenvolver um módulo no controlador de rede para coletar dados estatísticos dos fluxos da SDN, foi possível desenvolver um módulo interno no controlador Floodlight chamado *StatisticsCollector* que possibilitou a coleta precisa dos fluxos com recursos nativos do OpenFlow (mensagens assíncronas *Flow-Removed*) e que envia essas mensagens para uma plataforma de *streaming* distribuída (Apache Kafka) que, por sua vez, transmite em tempo real para os assinantes de seus tópicos.

O desenvolvimento do *SparkModule* foi fundamental para atingir o segundo e o terceiro objetivo. Através do componente Spark *Streaming* foi possível prover uma tarefa no *SparkModule* para armazenar estatísticas de fluxo da rede em tempo real em estrutura de *big data* NoSQL com o MongoDB. Além disso, foi possível desenvolver análises que utilizam aprendizado de máquina através do componente MLlib do Apache Spark de forma que possibilitem receber parâmetros flexíveis e agendamento de tarefas por meio da API REST desenvolvida.

Por fim, o desenvolvimento do módulo interno no controlador Floodlight chamado de *QoSManager* permitiu aplicar regras simples de QoS a perfis de tráfego resultantes das análises do *SparkModule*, usando limitação na taxa de transferência com recursos nativos do protocolo OpenFlow.

A prova de conceito inicial foi convincente quanto aos três conceitos importantes na solução que se deseja testar, quais foram, o funcionamento do "StatisticsCollector"; extração de valor com análise no *SparkModule*; e aplicação de politicas de QoS utilizando o "QoSManager". Isso permitiu o avanço na pesquisa, a qual, em seguida utilizou dados extraídos de uma rede real para extração de conhecimento com uma análise desenvolvida no *SparkModule*. Essa análise utilizou o algoritmo de aprendizado de máquina não supervisionado, k-means, para extrair três perfis de comportamentos de tráfego distintos analisando características dos fluxos como duração, quantidade de pacotes, quantidade de bytes e outras obtidas a partir dessas.

A análise conseguiu demonstrar que 77,78% dos fluxos analisados corresponderam a apenas 9,44% do volume total trafegado na rede com duração média era de 3,76 segundos, dando indícios de fluxos ratos no cenário analisado, segundo o que diz Vilela (2006). Nesse mesmo contexto, os fluxos responsáveis por 70,11% do volume trafegado corresponderam a apenas 10,98% do tráfego total e possuem duração média de 66,8 segundos, dando indícios de ser fluxos elefantes, conforme (ROCHA et al., 2017; SILVA et al., 2018). Os demais fluxos se mostraram bem estáveis quanto à duração, distribuição e volume.

Com a análise desenvolvida no *SparkModule*, é possível realizar uma investigação exploratória em cada *cluster*, dando uma visão detalhada do comportamento de cada um para auxiliar na administração dos recursos da rede. Com isso, regras de QoS podem ser associadas aos *clusters* e, por exemplo, limitar a taxa de transferência dos fluxos elefantes que possam estar interferindo no desempenho de fluxos que tenham maior relevância para um determinado cenário.

Considerando o curto tempo dessa pesquisa e a complexidade encontrada nos temas que foram abordados, os estudos de casos iniciais abrem muitas ideias interessantes sobre o uso dessas tecnologias combinadas para beneficiar o aproveitamento dos recursos de Rede Definida por Software multiserviço. Por meio deles, foi possível demonstrar a extração de conhecimento a partir das estatísticas de fluxos e aplicação de QoS simples utilizando recursos nativos do OpenFlow. No entanto, é preciso reconhecer que existem algumas limitações que precisam ser superadas e que trabalhos futuros podem enriquecer ainda mais os resultados iniciais.

6.1 Dificuldades e Limitações

A solução proposta nesse trabalho (DETCCS), inicialmente teve o intuito de utilizar apenas os recursos nativos do protocolo OpenFlow disponíveis pela biblioteca utilizada no controlador de rede Floodlight. No entanto, foi observado em outros trabalhos que o uso de protocolo como NetFlow e sFlow pode ser vantajoso para capturar dados de fluxos da rede por possuir maiores informações, sobretudo da camada de aplicação dos fluxos. Ademais, para aplicar políticas de QoS à adoção do OVSDB para criação e filas pode trazer boas contribuições. A decisão de utilizar apenas os recursos do OpenFlow pode ter limitado os resultados desse trabalho.

O cenário que foi utilizado nesta pesquisa é um cenário básico, conta com alguns hosts e *switchs* criados em emulador de rede e apenas um controlador. Esse cenário certamente é bem mais simples e menos complexo que o de uma rede em produção. Desta forma, acaba sendo um limite do trabalho.

Toda a estrutura utilizada durante o desenvolvimento desse trabalho foi a do ELAN. Desta forma, não obtivemos custos financeiros para aquisição de servidores para processar e armazenar dados. Em um cenário real, esse quesito é bastante importante e pode ser um limitador. Por esse motivo, sempre optamos por soluções de software sem custos para viabilizar a solução.

Uma das expectativas desse trabalho na implementação de políticas de QoS era de trabalhar com tabelas de medidores (*Meter Table*). No entanto, o OpenVSwitch não implementa todos os tipos de banda (*Meter Bands)*) dos medidores especificados pelo OpenFlow. Por exemplo, o *Dscp remark* ainda não é suportado. Essa funcionalidade, segundo Marques (2012), daria a possibilidade de remarcação automática do campo DSCP de acordo com a taxa de um determinado fluxo. Com esta ação, seria possível realizar políticas de QoS mais sofisticadas. Por exemplo, este seria o caso do serviço *Scavenger*, em que caso um fluxo ultrapasse a taxa delimitada por uma banda, este teria os pacotes marcados com um novo valor DSCP e seria encaminhado para uma fila de baixa prioridade.

6.2 Trabalhos Futuros

Embora Macedo et al. (2015) ensine que a seleção de atributos seja uma etapa bastante importante na fase de pré-processamento para selecionar o melhor subconjunto de atributos, reduzir a dimensão do conjunto e também ajudar na compreensão dos dados, além de reduzir a exigência de computação e melhorar o desempenho do modelo, nesse teste não utilizamos nenhuma técnica para selecionar os recursos, visto que são poucos os disponíveis pelo OpenFlow. Desta forma, há possibilidade da existência de atributos redundantes e incoerências que poderiam ter sido removidos do conjunto de dados. Em trabalhos futuros pretende-se utilizar técnicas específicas para esse propósito.

Em nossa solução, as regras de QoS criadas pelo administrador só são aplicadas aos fluxos que previamente foram classificados por uma análise. Esse é um dos principais pontos a serem melhorados e implementados nas próximas etapas desse trabalho. O artigo de (KURANAGE; PIAMRAT; HAMMA, 2019) pode servir de inspiração para implementação de técnicas de aprendizado de máquina supervisionado junto com aprendizado não supervisionado.

A criação de outras análises no *SparkModule* é um trabalho que pode ser constante e evolutivo. O olhar crítico para o cenário da rede pode despertar ao administrador o interesse de averiguar correlação entre varáveis diversas, que podem resultar em conhecimentos valorosos para auxiliar na administração dos recursos da SDN. Neste sentido, o trabalho de Jiang et al. (2010) pode ser norteador.

Com a API desenvolvida, ambientes para agendar tarefas, analisar os resultados obtidos com as análises do *SparkModule*, bem como a criação de políticas de QoS podem ser desenvolvidos para várias plataformas, a exemplo Web, mobile ou desktop.

A implementação de políticas de QoS implementadas nesse trabalho foram restritas a utilização de medidores do OpenFlow para limitar a taxa de transferência dos fluxos. No entanto, essas políticas podem ser utilizadas em conjunto com filas para definir largura de banda. Os trabalhos de Carvalho (2017) e Oliveira (2020), desenvolvidos no mesmo laboratório, ELAN, podem ser combinados com esse para atingir essa finalidade.

Referências

ALWASEL, K. et al. **Programming SDN-Native Big Data Applications: Research Gap Analysis**. *IEEE Cloud Computing*, v. 4, n. 5, p. 62–71, Sep. 2017. Citado 4 vezes nas páginas 16, 20, 45 e 57.

AMARAL, P. et al. **Machine learning in software defined networks: Data collection and traffic classification**. In: IEEE. *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. [S.l.], 2016. p. 1–5. Citado 8 vezes nas páginas 16, 48, 49, 50, 85, 94, 95 e 96.

AMBARI, A. ***Introduction***. 2020. Disponível em: <https://ambari.apache.org/>. Citado na página 81.

ANURADHA, J. et al. **A brief introduction on Big Data 5Vs characteristics and Hadoop technology**. *Procedia computer science*, Elsevier, v. 48, p. 319–324, 2015. Citado na página 40.

AZEVEDO, A. I. R. L.; SANTOS, M. F. **KDD, SEMMA and CRISP-DM: a parallel overview**. *IADS-DM*, 2008. Citado 4 vezes nas páginas 79, 85, 87 e 88.

BABIAR, J. et al. ***Configuration Guidelines for DiffServ Service Classes***. IETF, 2006. RFC 4594. (Request for Comments, 4594). Disponível em: <https://tools.ietf.org/html/rfc4594>. Citado na página 17.

BAKHSHI, T. **Multi-feature Enterprise Traffic Characterization in OpenFlow-based Software Defined Networks**. p. 23–28, Dec 2017. Citado 2 vezes nas páginas 47 e 48.

BAKHSHI, T.; GHITA, B. **User traffic profiling**. In: *2015 Internet Technologies and Applications (ITA)*. [S.l.: s.n.], 2015. p. 91–97. Citado 8 vezes nas páginas 15, 20, 45, 47, 83, 85, 87 e 96.

BEAN, R. ***Executives Report Measurable Results From Big Data, But Challenges Remain***. 2017. Disponível em: <https://www.forbes.com/sites/ciocentral/2017/01/10/executives-report-measurable-results-from-big-data-but-challenges-remain/#6aa279522287>. Citado na página 44.

BROWN, C. ***7 desafios que organizações enfrentam para extrair valor do Big Data***. 2019. Disponível em: <https://cio.com.br/7-desafios-que-organizacoes-enfrentam-para-extrair-valor-do-big-data/>. Citado na página 44.

CARVALHO, G. L. R. d. **Compartilhamento de banda em redes definidas por software**. Universidade Federal de Sergipe, 2017. Citado 2 vezes nas páginas 69 e 107.

CISCO. ***DiffServ – The Scalable End-to-End QoS Model***. 2005. Disponível em: <https://www.cisco.com/en/US/technologies/tk543/tk766/technologies_white_paper09186a00800a3e2f.html>. Citado na página 36.

CISCO. ***Perguntas mais freqüentes sobre QoS***. 2009. Disponível em: <https://www.cisco.com/c/pt_br/support/docs/quality-of-service-qos/qos-policing/22833-qos-faq.html>. Citado na página 35.

CISCO. ***Introduction to Cisco IOS NetFlow - A Technical Overview***. 2012. Disponível em: <https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html>. Citado na página 45.

CLAFFY, K. C. **Internet traffic characterization**. 1995. Citado 2 vezes nas páginas 15 e 47.

COMMUNITY, F. T. ***The Silhouette Coefficient***. 2020. Disponível em: <https://cs.fit.edu/~pkc/classes/ml-internet/silhouette.pdf>. Citado na página 96.

CUI, L.; YU, F. R.; YAN, Q. **When big data meets software-defined networking: SDN for big data and big data for SDN**. *IEEE network*, IEEE, v. 30, n. 1, p. 58–65, 2016. Citado 7 vezes nas páginas 14, 15, 16, 17, 21, 40 e 44.

D-ITG. ***D-ITG***. 2019. Disponível em: <http://www.grid.unina.it/software/ITG/>. Citado na página 84.

DATABRICKS. ***Apache Spark*** . 2020. Disponível em: <https://databricks.com/spark/about>. Citado 2 vezes nas páginas 73 e 74.

DESAI, A.; NAGEGOWDA, K. S.; NINIKRISHNA, T. **An approach to efficient network design and characterization using SDN and Hadoop**. In: *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*. [S.l.: s.n.], 2016. p. 1–6. Citado 5 vezes nas páginas 16, 18, 19, 45 e 57.

DESAI, A.; S, N. K. **Advanced Control Distributed Processing Architecture (ACDPA) using SDN and Hadoop for identifying the flow characteristics and setting the quality of service(QoS) in the network**. In: *2015 IEEE International Advance Computing Conference (IACC)*. [S.l.: s.n.], 2015. p. 784–788. Citado 6 vezes nas páginas 15, 16, 18, 19, 45 e 57.

DESAI, A.; S, N. K.; T, N. **Secure and QoS aware architecture for cloud using software defined networks and Hadoop**. In: *2015 International Conference on Computing and Network Communications (CoCoNet)*. [S.l.: s.n.], 2015. p. 369–373. Citado 5 vezes nas páginas 16, 18, 19, 45 e 57.

DICIO. ***Dicionário Online de Português***. 2020. Disponível em: <https://www.dicio.com.br/>. Citado na página 67.

FAN, Z.; LIU, R. **Investigation of machine learning based network traffic classification**. p. 1–6, Aug 2017. ISSN 2154-0225. Citado 4 vezes nas páginas 16, 48, 49 e 50.

FERREIRA, C. C. et al. **Análise comparativa de controladores para redes definidas por software de Classe Carrier Grade**. Universidade Federal de Uberlândia, 2016. Citado na página 68.

FISHER, D. et al. **Interactions with big data analytics, interactions, v. 19 n. 3**. *May+ June*, 2012. Citado na página 39.

FLOODLIGHT. ***Floodlight***. 2019. Disponível em: <http://www.projectfloodlight.org/floodlight/>. Citado 2 vezes nas páginas 68 e 84.

FOUNDATION, O. N. F. ***OpenFlow Switch Specification***. 2012. Disponível em: <https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.3.1.pdf>. Citado 7 vezes nas páginas 31, 33, 34, 45, 46, 70 e 101.

FRäNTI, P.; SIERANOJA, S. **How much can k-means be improved by using better initialization and repeats?** *Pattern Recognition*, v. 93, p. 95 – 112, 2019. ISSN 0031-3203. Citado na página 50.

GIL, A. C. ***Métodos e técnicas de pesquisa social***. [S.l.]: 4. ed. Ediitora Atlas SA, 2002. Citado na página 23.

GUEDES, D. et al. **Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores**. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*, v. 30, n. 4, p. 160–210, 2012. Citado na página 31.

HAYES, M. et al. **Scalable Architecture for SDN Traffic Classification**. *IEEE Systems Journal*, v. 12, n. 4, p. 3203–3214, 2018. Citado na página 48.

HU, F.; HAO, Q.; BAO, K. **A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation**. *IEEE Communications Surveys Tutorials*, v. 16, n. 4, p. 2181–2206, Fourthquarter 2014. ISSN 1553-877X. Citado na página 31.

HU, H. et al. **Toward Scalable Systems for Big Data Analytics: A Technology Tutorial**. *IEEE Access*, v. 2, p. 652–687, 2014. Citado 3 vezes nas páginas 41, 42 e 44.

HU, W. et al. An efficient transportation architecture for big data movement. In: *2013 9th International Conference on Information, Communications Signal Processing*. [S.l.: s.n.], 2013. p. 1–5. Citado na página 39.

ISO. ***Information technology — Quality of service: Framework***. Geneva, CH, 1998. Citado na página 34.

JAIN, S. et al. **Applying big data technologies to manage QoS in an SDN**. In: *2016 12th International Conference on Network and Service Management (CNSM)*. [S.l.: s.n.], 2016. p. 302–306. Citado 5 vezes nas páginas 15, 16, 19, 45 e 57.

JIANG, H. et al. **Network prefix-level traffic profiling: Characterizing, modeling, and evaluation**. *Computer Networks*, v. 54, n. 18, p. 3327 – 3340, 2010. ISSN 1389-1286. Citado 6 vezes nas páginas 45, 47, 85, 87, 96 e 107.

JIN, X. et al. **Significance and Challenges of Big Data Research**. *Big Data Research*, v. 2, n. 2, p. 59 – 64, 2015. ISSN 2214-5796. Visions on Big Data. Citado 2 vezes nas páginas 39 e 40.

JOSE, A. S.; NAIR, L. R.; PAUL, V. **Data mining in software defined networking - a survey**. In: *2017 International Conference on Computing Methodologies and Communication (ICCMC)*. [S.l.: s.n.], 2017. p. 668–672. Citado 3 vezes nas páginas 18, 45 e 57.

KAFKA, A. ***Apache Kafka***. 2020. Disponível em: <https://kafka.apache.org/intro>. Citado na página 73.

KAMIENSKI, C. A.; SADOK, D. **Qualidade de serviço na internet**. *minicurso, 18o SBRC, Belo Horizonte/MG*, 2000. Citado na página 36.

KANUNGO, T. et al. **An efficient k-means clustering algorithm: analysis and implementation**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 7, p. 881–892, 2002. Citado na página 50.

KARAKUS, M.; DURRESI, A. *Quality of service (QoS) in software defined networking (SDN): A survey*. *Journal of Network and Computer Applications*, Elsevier, v. 80, p. 200–218, 2017. Citado 5 vezes nas páginas 14, 15, 34, 38 e 39.

KAUR, N.; SOOD, S. **Dynamic resource allocation for big data streams based on data characteristics (5Vs)**. *International Journal of Network Management*, v. 27, n. 4, 2017. Cited By 3. Citado 3 vezes nas páginas 40, 41 e 42.

KENDIG, C. E. **What is proof of concept research and how does it generate epistemic and ethical categories for future scientific practice?** *Science and Engineering Ethics*, Springer, v. 22, n. 3, p. 735–753, 2016. Citado na página 83.

KHATER, A.; HASHEMI, M. R. **Dynamic Flow Management Based on DiffServ in SDN Networks**. In: *Electrical Engineering (ICEE), Iranian Conference on*. [S.l.: s.n.], 2018. p. 1505–1510. Citado na página 16.

KITCHENHAM, B.; CHARTERS, S. ***Guidelines for performing systematic literature reviews in software engineering***. [S.l.], 2007. Citado 2 vezes nas páginas 51 e 65.

KOLOMVATSOS, K. et al. **Uncertainty-driven ensemble forecasting of QoS in Software Defined Networks**. In: *2017 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.: s.n.], 2017. p. 1284–1289. Citado 2 vezes nas páginas 45 e 57.

KREUTZ, D. et al. **Software-Defined Networking: A Comprehensive Survey**. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 0018-9219. Citado 3 vezes nas páginas 15, 25 e 32.

KUANG, L. et al. **A tensor-based big data model for QoS improvement in software defined networks**. *IEEE Network*, v. 30, n. 1, p. 30–35, January 2016. Citado na página 57.

KUNE, R. et al. **The anatomy of big data computing**. *Software: Practice and Experience*, Wiley Online Library, v. 46, n. 1, p. 79–105, 2016. Citado 3 vezes nas páginas 14, 52 e 61.

KURANAGE, M. P. J.; PIAMRAT, K.; HAMMA, S. **Network Traffic Classification Using Machine Learning for Software Defined Networks**. In: SPRINGER. *International Conference on Machine Learning for Networking*. [S.l.], 2019. p. 28–39. Citado 6 vezes nas páginas 15, 20, 94, 95, 96 e 107.

LANEY, D. 3d data management: Controlling data volume, velocity and variety. *META group research note*, v. 6, n. 70, p. 1, 2001. Citado na página 39.

LARA, A.; KOLASANI, A.; RAMAMURTHY, B. **Network Innovation using OpenFlow: A Survey**. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 493–512, First 2014. ISSN 1553-877X. Citado na página 31.

LE, L. et al. **SDN/NFV, Machine Learning, and Big Data Driven Network Slicing for 5G**. In: *2018 IEEE 5G World Forum (5GWF)*. [S.l.: s.n.], 2018. p. 20–25. Citado 2 vezes nas páginas 20 e 91.

LEARN scikit. ***Selecting the number of clusters with silhouette analysis on KMeans clustering***. 2020. Disponível em: <https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html>. Citado na página 96.

LOBITSKY, G. ***Big Data e Machine Learning estão no centro da transformação digital***. 2017. Disponível em: <https://cio.com.br/tendencias/big-data-e-machine-learning-estao-no-centro-da-transformacao-digital/>. Citado na página 15.

MACEDO, D. C. d. et al. **Método de redução de dimensionalidade de dados derivados do domínio de expressão gênica**. Universidade Tecnológica Federal do Paraná, 2015. Citado na página 106.

MACEDO, D. F. et al. **Programmable networks—From software-defined radio to software-defined networking**. *IEEE communications surveys & tutorials*, IEEE, v. 17, n. 2, p. 1102–1125, 2015. Citado 4 vezes nas páginas 28, 29, 30 e 68.

MANTUR, B.; DESAI, A.; NAGEGOWDA, K. **Centralized control signature-based firewall and statistical-based network intrusion detection system (NIDS) in software defined networks (SDN)**. In: *Emerging Research in Computing, Information, Communication and Applications*. [S.l.]: Springer, 2015. p. 497–506. Citado na página 19.

MANYIKA, J. **Big data: The next frontier for innovation, competition, and productivity**. *http://www. mckinsey. com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation*, 2011. Citado na página 39.

MARCONI, M. d. A.; LAKATOS, E. M. **Fundamentos de metodologia cientifica. 5 edição**. *Rio de Janeiro: Editora Atlas*, 2003. Citado na página 23.

MARQUES, D. d. A. L. ***Um estudo sobre a aplicação de uma rede de circuitos dinâmicos em domínios openflow***. Dissertação (Mestrado), 2012. Citado na página 106.

MARR, B. ***Building Your Big Data Infrastructure: 4 Key Components Every Business Needs To Consider***. 2019. Disponível em: <https://www.forbes.com/sites/bernardmarr/2016/06/15/building-your-big-data-infrastructure-4-key-components-every-business-needs-to-consider/#6dc06b5a577e>. Citado 2 vezes nas páginas 41 e 42.

MAYER-SCHÖNBERGER, V.; CUKIER, K. *Big data: A revolution that will transform how we live, work, and think*. [S.l.]: Houghton Mifflin Harcourt, 2013. Citado na página 39.

MCAFEE, A. et al. **Big data: the management revolution**. *Harvard business review*, v. 90, n. 10, p. 60–68, 2012. Citado na página 39.

MCKEOWN, N. et al. **OpenFlow: enabling innovation in campus networks**. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 2, p. 69–74, 2008. Citado 2 vezes nas páginas 31 e 32.

MENDONCA, R. ***Arquiteturas de QoS – Parte 02***. 2014. Disponível em: <http://blog.ccna.com.br/2014/04/30/arquiteturas-de-qos-parte-02/>. Citado 2 vezes nas páginas 37 e 38.

MININET. ***Mininet***. 2019. Disponível em: <http://mininet.org/>. Citado na página 84.

MOHANTY, H.; BHUYAN, P.; CHENTHATI, D. ***Big data: A primer***. [S.l.]: Springer, 2015. v. 11. Citado 2 vezes nas páginas 43 e 44.

NASCIMENTO, F. P. d. N.; SOUSA, F. L. L. ***Metodologia da pesquisa científica teoria e prática: como elaborar TCC***. [S.l.]: Fortaleza: INESP, 2017. Citado 2 vezes nas páginas 22 e 23.

NICHOLS, K. et al. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. IETF, 1998. RFC 2474. (Request for Comments, 2474). Disponível em: <https://tools.ietf.org/html/rfc2474>. Citado na página 17.

NUNES, B. A. A. et al. **A survey of software-defined networking: Past, present, and future of programmable networks**. *IEEE Communications Surveys & Tutorials*, IEEE, v. 16, n. 3, p. 1617–1634, 2014. Citado 7 vezes nas páginas 25, 26, 29, 30, 31, 32 e 68.

OLIVEIRA, J. A. M. d. **Modelos de Alocação de Banda com Justiça em Controladores de Redes Definidas por Software**. Universidade Federal de Sergipe, 2020. Citado na página 107.

ONF. ***Open Network Foundation ONF***. 2019. Disponível em: <https://www.opennetworking.org>. Citado 2 vezes nas páginas 26 e 33.

OPENVSWITCH. ***Open vSwitch Open vSwitch***. 2019. Disponível em: <https://www.openvswitch.org>. Citado 2 vezes nas páginas 31 e 84.

PASSOS, A. et al. **A Systematic Mapping Study on Software Comments Analysis**. In: *SEKE*. [S.l.: s.n.], 2018. p. 681–680. Citado na página 52.

PENCHIKALA, S. ***Big Data com Apache Spark - Parte 1: Introdução***. 2020. Disponível em: <https://www.infoq.com/br/articles/apache-spark-introduction>. Citado na página 75.

PONNAPPAN, A. et al. **A policy based QoS management system for the IntServ/DiffServ based Internet**. p. 159–168, June 2002. Citado 3 vezes nas páginas 17, 36 e 37.

PRETORIUS, R.; BUDGEN, D. **A mapping study on empirical evidence related to the models and forms used in the uml**. In: ACM. *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. [S.l.], 2008. p. 342–344. Citado na página 65.

PRODANOV, C. C.; FREITAS, E. C. de. ***Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição***. [S.l.]: Editora Feevale, 2013. Citado na página 21.

PROVOST, F.; FAWCETT, T. **Data science and its relationship to big data and data-driven decision making**. *Big data*, Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA, v. 1, n. 1, p. 51–59, 2013. Citado na página 39.

ROCHA, A. L. B. et al. **EFM-Elephant Flow Manager: uma arquitetura para detecção e tratamento de fluxos elefantes em DCNs**. Universidade Federal de São Carlos, 2017. Citado 2 vezes nas páginas 99 e 105.

ROJAS, J. S. ***IP Network Traffic Flows Labeled with 75 Apps***. 2019. Disponível em: <https://www.kaggle.com/jsrojas/ip-network-traffic-flows-labeled-with-87-apps>. Citado na página 95.

ROTHENBERG, C. E. et al. **OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes**. *Cad. CPqD Tecnologia, Campinas*, v. 7, n. 1, p. 65–76, 2010. Citado na página 25.

SABATA, B. et al. **Taxonomy for QoS specifications**. In: IEEE. *Proceedings Third International Workshop on Object-Oriented Real-Time Dependable Systems*. [S.l.], 1997. p. 100–107. Citado 3 vezes nas páginas 35, 53 e 62.

SANDRI, S. D. **DATA MINING–APLICAÇÃO EM UMA BASE DE DADOS REAL COM DADOS DE USUÁRIOS DOS FARÓIS DO SABER**. *Revista Eletrônica de Sistemas de Informação*, v. 7, n. 2, 2008. Citado na página 80.

SANTANA, F. ***Entenda o Algoritmo K-means e Saiba como Aplicar essa Técnica***. 2019. Disponível em: <https://minerandodados.com.br/entenda-o-algoritmo-k-means/>. Citado na página 50.

SEMENCIUC, E. et al. **Performance evaluation of a Cloud-based QoS support mechanism**. In: *2016 International Conference on Communications (COMM)*. [S.l.: s.n.], 2016. p. 243–246. Citado 3 vezes nas páginas 18, 45 e 57.

SILVA, A. S. D. et al. **Identification and Selection of Flow Features for Accurate Traffic Classification in SDN**. In: *2015 IEEE 14th International Symposium on Network Computing and Applications*. [S.l.: s.n.], 2015. p. 134–141. Citado na página 95.

SILVA, E. F. d. et al. **SDNMonitor: um serviço de monitoramento de tráfego em redes definidas por software**. Universidade Federal de Sergipe, 2016. Citado 2 vezes nas páginas 45 e 69.

SILVA, M. V. B. da et al. **Identificação de Fluxos Elefantes em Redes de Ponto de Troca de Tráfego com Suporteá Programabilidade P4**. In: SBC. *Anais Principais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.], 2018. Citado 3 vezes nas páginas 98, 99 e 105.

SILVA, R. P. B. et al. **Metodologia de caracterização e modelagem de tráfego para transmissão de imagens médicas**. Universidade Federal de Sergipe, 2015. Citado 2 vezes nas páginas 48 e 94.

SPARK, A. ***Apache Spark***. 2020. Disponível em: <https://spark.apache.org>. Citado 2 vezes nas páginas 73 e 74.

STATISTA. ***Most famous social network sites worldwide as of July 2019, ranked by number of active users (in millions)***. 2019. Disponível em: <https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/>. Citado na página 43.

STATS, I. L. ***Internet Live Stats***. 2019. Disponível em: <https://www.internetlivestats.com/>. Citado na página 40.

SYAKUR, M. et al. **Integration k-means clustering method and elbow method for identification of the best customer profile cluster**. In: IOP PUBLISHING. *IOP Conference Series: Materials Science and Engineering*. [S.l.], 2018. v. 336, n. 1, p. 012017. Citado na página 96.

TEITELBAUM, J.; HANSS. ***QoS Requirements for Internet2 (draft)***. 1998. Disponível em: <http://exodus.cs.ccu.edu.tw/~rhhwang/RFC/requirements.html>. Citado na página 35.

TRENDS, G. ***Veja o que o mundo está pesquisando***. 2020. Disponível em: <https://trends.google.com.br/>. Citado na página 15.

VIEIRA, A. F. de C. ***Otimização do Processo de Laminação a Frio através de planejamentos de Experimentos***. Tese (Doutorado) — PUC-Rio, 2007. Citado na página 79.

VILELA, G. S. **Caracterizaçao de Tráfego Utilizando Classificaçao de Fluxos de Comunicaçao**. *Mestre em Ciências em Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil*, 2006. Citado 3 vezes nas páginas 48, 98 e 105.

WALLNER, R. ***How to implement Quality Of Service using Floodlight***. 2013. Disponível em: <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343504/How+to+implement+Quality+Of+Service+using+Floodlight?showComments=true&showCommentArea=true>. Citado 2 vezes nas páginas 16 e 17.

XIA, W. et al. **A survey on software-defined networking**. *IEEE Communications Surveys & Tutorials*, IEEE, v. 17, n. 1, p. 27–51, 2015. Citado 5 vezes nas páginas 26, 27, 28, 29 e 30.

YIN, R. **Estudo de Caso-: Planejamento e Métodos.[sl] Bookman editora**. 2015. Citado 2 vezes nas páginas 23 e 79.

YOUNAS, M. **Research challenges of big data**. *Service Oriented Computing and Applications*, v. 13, n. 2, p. 105–107, Jun 2019. ISSN 1863-2394. Citado 2 vezes nas páginas 40 e 41.